



Asterisk 12 Reference

Asterisk Development Team <asteriskteam@digium.com>

1. New in 12	14
2. Upgrading to Asterisk 12	28
3. Asterisk 12 Installation and Configuration	35
3.1 Installing pjproject	36
3.2 Configuring res_pjsip	40
3.2.1 Configuring res_pjsip to work through NAT	54
3.2.2 Setting up PJSIP Realtime	58
3.3 Managing Realtime Databases with Alembic	63
4. Getting Started with ARI	65
4.1 Create a new resource with ARI	74
5. Asterisk 12 Specifications	78
5.1 AMI v2 Specification	79
5.2 Asterisk 12 CDR Specification	100
5.3 Asterisk 12 CEL Specification	122
6. Asterisk 12 Command Reference	133
6.1 Asterisk 12 AGI Commands	134
6.1.1 Asterisk 12 AGICommand_answer	135
6.1.2 Asterisk 12 AGICommand_asyncagi break	136
6.1.3 Asterisk 12 AGICommand_channel status	137
6.1.4 Asterisk 12 AGICommand_control stream file	138
6.1.5 Asterisk 12 AGICommand_database del	139
6.1.6 Asterisk 12 AGICommand_database deltree	140
6.1.7 Asterisk 12 AGICommand_database get	141
6.1.8 Asterisk 12 AGICommand_database put	142
6.1.9 Asterisk 12 AGICommand_exec	143
6.1.10 Asterisk 12 AGICommand_get data	144
6.1.11 Asterisk 12 AGICommand_get full variable	145
6.1.12 Asterisk 12 AGICommand_get option	146
6.1.13 Asterisk 12 AGICommand_get variable	147
6.1.14 Asterisk 12 AGICommand_gosub	148
6.1.15 Asterisk 12 AGICommand_hangup	149
6.1.16 Asterisk 12 AGICommand_noop	150
6.1.17 Asterisk 12 AGICommand_receive char	151
6.1.18 Asterisk 12 AGICommand_receive text	152
6.1.19 Asterisk 12 AGICommand_record file	153
6.1.20 Asterisk 12 AGICommand_say alpha	154
6.1.21 Asterisk 12 AGICommand_say date	155
6.1.22 Asterisk 12 AGICommand_say datetime	156
6.1.23 Asterisk 12 AGICommand_say digits	157
6.1.24 Asterisk 12 AGICommand_say number	158
6.1.25 Asterisk 12 AGICommand_say phonetic	159
6.1.26 Asterisk 12 AGICommand_say time	160
6.1.27 Asterisk 12 AGICommand_send image	161
6.1.28 Asterisk 12 AGICommand_send text	162
6.1.29 Asterisk 12 AGICommand_set autohangup	163
6.1.30 Asterisk 12 AGICommand_set callerid	164
6.1.31 Asterisk 12 AGICommand_set context	165
6.1.32 Asterisk 12 AGICommand_set extension	166
6.1.33 Asterisk 12 AGICommand_set music	167
6.1.34 Asterisk 12 AGICommand_set priority	168
6.1.35 Asterisk 12 AGICommand_set variable	169
6.1.36 Asterisk 12 AGICommand_speech activate grammar	170
6.1.37 Asterisk 12 AGICommand_speech create	171
6.1.38 Asterisk 12 AGICommand_speech deactivate grammar	172
6.1.39 Asterisk 12 AGICommand_speech destroy	173
6.1.40 Asterisk 12 AGICommand_speech load grammar	174
6.1.41 Asterisk 12 AGICommand_speech recognize	175
6.1.42 Asterisk 12 AGICommand_speech set	176
6.1.43 Asterisk 12 AGICommand_speech unload grammar	177
6.1.44 Asterisk 12 AGICommand_stream file	178
6.1.45 Asterisk 12 AGICommand_tdd mode	179

6.1.46 Asterisk 12 AGICommand_verbose	180
6.1.47 Asterisk 12 AGICommand_wait for digit	181
6.2 Asterisk 12 AMI Actions	182
6.2.1 Asterisk 12 ManagerAction_AbsoluteTimeout	183
6.2.2 Asterisk 12 ManagerAction_AgentLogoff	184
6.2.3 Asterisk 12 ManagerAction_Agents	185
6.2.4 Asterisk 12 ManagerAction_AGI	186
6.2.5 Asterisk 12 ManagerAction_AOCMessage	187
6.2.6 Asterisk 12 ManagerAction_Atxfer	189
6.2.7 Asterisk 12 ManagerAction_BlindTransfer	190
6.2.8 Asterisk 12 ManagerAction_Bridge	191
6.2.9 Asterisk 12 ManagerAction_BridgeDestroy	192
6.2.10 Asterisk 12 ManagerAction_BridgeInfo	193
6.2.11 Asterisk 12 ManagerAction_BridgeKick	194
6.2.12 Asterisk 12 ManagerAction_BridgeList	195
6.2.13 Asterisk 12 ManagerAction_BridgeTechnologyList	196
6.2.14 Asterisk 12 ManagerAction_BridgeTechnologySuspend	197
6.2.15 Asterisk 12 ManagerAction_BridgeTechnologyUnsuspend	198
6.2.16 Asterisk 12 ManagerAction_Challenge	199
6.2.17 Asterisk 12 ManagerAction_ChangeMonitor	200
6.2.18 Asterisk 12 ManagerAction_Command	201
6.2.19 Asterisk 12 ManagerAction_ConfbridgeKick	202
6.2.20 Asterisk 12 ManagerAction_ConfbridgeList	203
6.2.21 Asterisk 12 ManagerAction_ConfbridgeListRooms	204
6.2.22 Asterisk 12 ManagerAction_ConfbridgeLock	205
6.2.23 Asterisk 12 ManagerAction_ConfbridgeMute	206
6.2.24 Asterisk 12 ManagerAction_ConfbridgeSetSingleVideoSrc	207
6.2.25 Asterisk 12 ManagerAction_ConfbridgeStartRecord	208
6.2.26 Asterisk 12 ManagerAction_ConfbridgeStopRecord	209
6.2.27 Asterisk 12 ManagerAction_ConfbridgeUnlock	210
6.2.28 Asterisk 12 ManagerAction_ConfbridgeUnmute	211
6.2.29 Asterisk 12 ManagerAction_ControlPlayback	212
6.2.30 Asterisk 12 ManagerAction_CoreSettings	213
6.2.31 Asterisk 12 ManagerAction_CoreShowChannels	214
6.2.32 Asterisk 12 ManagerAction_CoreStatus	215
6.2.33 Asterisk 12 ManagerAction_CreateConfig	216
6.2.34 Asterisk 12 ManagerAction_DAHDI DialOffhook	217
6.2.35 Asterisk 12 ManagerAction_DAHDI DNDoff	218
6.2.36 Asterisk 12 ManagerAction_DAHDI DNDon	219
6.2.37 Asterisk 12 ManagerAction_DAHDI Hangup	220
6.2.38 Asterisk 12 ManagerAction_DAHDI Restart	221
6.2.39 Asterisk 12 ManagerAction_DAHDI ShowChannels	222
6.2.40 Asterisk 12 ManagerAction_DAHDI Transfer	223
6.2.41 Asterisk 12 ManagerAction_DataGet	224
6.2.42 Asterisk 12 ManagerAction_DBDel	225
6.2.43 Asterisk 12 ManagerAction_DBDelTree	226
6.2.44 Asterisk 12 ManagerAction_DBGet	227
6.2.45 Asterisk 12 ManagerAction_DBPut	228
6.2.46 Asterisk 12 ManagerAction_Events	229
6.2.47 Asterisk 12 ManagerAction_ExtensionState	230
6.2.48 Asterisk 12 ManagerAction_Filter	231
6.2.49 Asterisk 12 ManagerAction_FilterList	232
6.2.50 Asterisk 12 ManagerAction_GetConfig	233
6.2.51 Asterisk 12 ManagerAction_GetConfigJSON	234
6.2.52 Asterisk 12 ManagerAction_Getvar	235
6.2.53 Asterisk 12 ManagerAction_Hangup	236
6.2.54 Asterisk 12 ManagerAction_IAXnetstats	237
6.2.55 Asterisk 12 ManagerAction_IAXpeerlist	238
6.2.56 Asterisk 12 ManagerAction_IAXpeers	239
6.2.57 Asterisk 12 ManagerAction_IAXregistry	240
6.2.58 Asterisk 12 ManagerAction_JabberSend_res_jabber	241

6.2.59 Asterisk 12 ManagerAction_JabberSend_res_xmpp	242
6.2.60 Asterisk 12 ManagerAction_ListCategories	243
6.2.61 Asterisk 12 ManagerAction_ListCommands	244
6.2.62 Asterisk 12 ManagerAction_LocalOptimizeAway	245
6.2.63 Asterisk 12 ManagerAction_Login	246
6.2.64 Asterisk 12 ManagerAction_Logoff	247
6.2.65 Asterisk 12 ManagerAction_MailboxCount	248
6.2.66 Asterisk 12 ManagerAction_MailboxStatus	249
6.2.67 Asterisk 12 ManagerAction_MeetmeList	250
6.2.68 Asterisk 12 ManagerAction_MeetmeListRooms	251
6.2.69 Asterisk 12 ManagerAction_MeetmeMute	252
6.2.70 Asterisk 12 ManagerAction_MeetmeUnmute	253
6.2.71 Asterisk 12 ManagerAction_MessageSend	254
6.2.72 Asterisk 12 ManagerAction_MixMonitor	255
6.2.73 Asterisk 12 ManagerAction_MixMonitorMute	256
6.2.74 Asterisk 12 ManagerAction_ModuleCheck	257
6.2.75 Asterisk 12 ManagerAction_ModuleLoad	258
6.2.76 Asterisk 12 ManagerAction_Monitor	259
6.2.77 Asterisk 12 ManagerAction_MuteAudio	260
6.2.78 Asterisk 12 ManagerAction_Originate	261
6.2.79 Asterisk 12 ManagerAction_Park	262
6.2.80 Asterisk 12 ManagerAction_ParkedCalls	263
6.2.81 Asterisk 12 ManagerAction_Parkinglots	264
6.2.82 Asterisk 12 ManagerAction_PauseMonitor	265
6.2.83 Asterisk 12 ManagerAction_Ping	266
6.2.84 Asterisk 12 ManagerAction_PJSIPNotify	267
6.2.85 Asterisk 12 ManagerAction_PJSIPQualify	268
6.2.86 Asterisk 12 ManagerAction_PJSIPShowEndpoint	269
6.2.87 Asterisk 12 ManagerAction_PJSIPShowEndpoints	270
6.2.88 Asterisk 12 ManagerAction_PJSIPShowRegistrationsInbound	271
6.2.89 Asterisk 12 ManagerAction_PJSIPShowRegistrationsOutbound	272
6.2.90 Asterisk 12 ManagerAction_PJSIPShowSubscriptionsInbound	273
6.2.91 Asterisk 12 ManagerAction_PJSIPShowSubscriptionsOutbound	274
6.2.92 Asterisk 12 ManagerAction_PJSIPUnregister	275
6.2.93 Asterisk 12 ManagerAction_PlayDTMF	276
6.2.94 Asterisk 12 ManagerAction_PresenceState	277
6.2.95 Asterisk 12 ManagerAction_PRIShowSpans	278
6.2.96 Asterisk 12 ManagerAction_QueueAdd	279
6.2.97 Asterisk 12 ManagerAction_QueueLog	280
6.2.98 Asterisk 12 ManagerAction_QueueMemberRingInUse	281
6.2.99 Asterisk 12 ManagerAction_QueuePause	282
6.2.100 Asterisk 12 ManagerAction_QueuePenalty	283
6.2.101 Asterisk 12 ManagerAction_QueueReload	284
6.2.102 Asterisk 12 ManagerAction_QueueRemove	285
6.2.103 Asterisk 12 ManagerAction_QueueReset	286
6.2.104 Asterisk 12 ManagerAction_QueueRule	287
6.2.105 Asterisk 12 ManagerAction_Queues	288
6.2.106 Asterisk 12 ManagerAction_QueueStatus	289
6.2.107 Asterisk 12 ManagerAction_QueueSummary	290
6.2.108 Asterisk 12 ManagerAction_Redirect	291
6.2.109 Asterisk 12 ManagerAction_Reload	292
6.2.110 Asterisk 12 ManagerAction_SendText	293
6.2.111 Asterisk 12 ManagerAction_Setvar	294
6.2.112 Asterisk 12 ManagerAction_ShowDialPlan	295
6.2.113 Asterisk 12 ManagerAction_SIPnotify	296
6.2.114 Asterisk 12 ManagerAction_SIPpeers	297
6.2.115 Asterisk 12 ManagerAction_SIPpeerstatus	298
6.2.116 Asterisk 12 ManagerAction_SIPqualifypeer	299
6.2.117 Asterisk 12 ManagerAction_SIPshowpeer	300
6.2.118 Asterisk 12 ManagerAction_SIPshowregistry	301
6.2.119 Asterisk 12 ManagerAction_SKINNYdevices	302

6.2.120 Asterisk 12 ManagerAction_SKINNYlines	303
6.2.121 Asterisk 12 ManagerAction_SKINNYshowdevice	304
6.2.122 Asterisk 12 ManagerAction_SKINNYshowline	305
6.2.123 Asterisk 12 ManagerAction_Status	306
6.2.124 Asterisk 12 ManagerAction_StopMixMonitor	307
6.2.125 Asterisk 12 ManagerAction_StopMonitor	308
6.2.126 Asterisk 12 ManagerAction_UnpauseMonitor	309
6.2.127 Asterisk 12 ManagerAction_UpdateConfig	310
6.2.128 Asterisk 12 ManagerAction_UserEvent	311
6.2.129 Asterisk 12 ManagerAction_VoicemailRefresh	312
6.2.130 Asterisk 12 ManagerAction_VoicemailUsersList	313
6.2.131 Asterisk 12 ManagerAction_WaitEvent	314
6.3 Asterisk 12 AMI Events	315
6.3.1 Asterisk 12 ManagerEvent_AgentCalled	316
6.3.2 Asterisk 12 ManagerEvent_AgentComplete	318
6.3.3 Asterisk 12 ManagerEvent_AgentConnect	320
6.3.4 Asterisk 12 ManagerEvent_AgentDump	322
6.3.5 Asterisk 12 ManagerEvent_AgentLogin	324
6.3.6 Asterisk 12 ManagerEvent_AgentLogoff	325
6.3.7 Asterisk 12 ManagerEvent_AgentRingNoAnswer	326
6.3.8 Asterisk 12 ManagerEvent_Agents	328
6.3.9 Asterisk 12 ManagerEvent_AgentsComplete	330
6.3.10 Asterisk 12 ManagerEvent_AGIExecEnd	331
6.3.11 Asterisk 12 ManagerEvent_AGIExecStart	332
6.3.12 Asterisk 12 ManagerEvent_Alarm	333
6.3.13 Asterisk 12 ManagerEvent_AlarmClear	334
6.3.14 Asterisk 12 ManagerEvent_AOC-D	335
6.3.15 Asterisk 12 ManagerEvent_AOC-E	337
6.3.16 Asterisk 12 ManagerEvent_AOC-S	339
6.3.17 Asterisk 12 ManagerEvent_AsyncAGIEnd	341
6.3.18 Asterisk 12 ManagerEvent_AsyncAGIExec	342
6.3.19 Asterisk 12 ManagerEvent_AsyncAGIStart	343
6.3.20 Asterisk 12 ManagerEvent_AttendedTransfer	344
6.3.21 Asterisk 12 ManagerEvent_BlindTransfer	349
6.3.22 Asterisk 12 ManagerEvent_BridgeCreate	351
6.3.23 Asterisk 12 ManagerEvent_BridgeDestroy	352
6.3.24 Asterisk 12 ManagerEvent_BridgeEnter	353
6.3.25 Asterisk 12 ManagerEvent_BridgeLeave	355
6.3.26 Asterisk 12 ManagerEvent_BridgeMerge	357
6.3.27 Asterisk 12 ManagerEvent_ChanSpyStart	358
6.3.28 Asterisk 12 ManagerEvent_ChanSpyStop	360
6.3.29 Asterisk 12 ManagerEvent_ConfbridgeEnd	362
6.3.30 Asterisk 12 ManagerEvent_ConfbridgeJoin	363
6.3.31 Asterisk 12 ManagerEvent_ConfbridgeLeave	365
6.3.32 Asterisk 12 ManagerEvent_ConfbridgeMute	367
6.3.33 Asterisk 12 ManagerEvent_ConfbridgeRecord	369
6.3.34 Asterisk 12 ManagerEvent_ConfbridgeStart	370
6.3.35 Asterisk 12 ManagerEvent_ConfbridgeStopRecord	371
6.3.36 Asterisk 12 ManagerEvent_ConfbridgeTalking	372
6.3.37 Asterisk 12 ManagerEvent_ConfbridgeUnmute	374
6.3.38 Asterisk 12 ManagerEvent_DAHDIChannel	376
6.3.39 Asterisk 12 ManagerEvent_DialBegin	377
6.3.40 Asterisk 12 ManagerEvent_DialEnd	379
6.3.41 Asterisk 12 ManagerEvent_DNDState	381
6.3.42 Asterisk 12 ManagerEvent_DTMFBegin	382
6.3.43 Asterisk 12 ManagerEvent_DTMFEnd	383
6.3.44 Asterisk 12 ManagerEvent_ExtensionStatus	384
6.3.45 Asterisk 12 ManagerEvent_FAXStatus	385
6.3.46 Asterisk 12 ManagerEvent_FullyBooted	387
6.3.47 Asterisk 12 ManagerEvent_Hangup	388
6.3.48 Asterisk 12 ManagerEvent_HangupHandlerPop	389

6.3.49 Asterisk 12 ManagerEvent_HangupHandlerPush	390
6.3.50 Asterisk 12 ManagerEvent_HangupHandlerRun	391
6.3.51 Asterisk 12 ManagerEvent_HangupRequest	392
6.3.52 Asterisk 12 ManagerEvent_Hold	393
6.3.53 Asterisk 12 ManagerEvent_LocalBridge	394
6.3.54 Asterisk 12 ManagerEvent_LocalOptimizationBegin	396
6.3.55 Asterisk 12 ManagerEvent_LocalOptimizationEnd	399
6.3.56 Asterisk 12 ManagerEvent_LogChannel	401
6.3.57 Asterisk 12 ManagerEvent_MCID	402
6.3.58 Asterisk 12 ManagerEvent_MeetmeEnd	404
6.3.59 Asterisk 12 ManagerEvent_MeetmeJoin	405
6.3.60 Asterisk 12 ManagerEvent_MeetmeLeave	406
6.3.61 Asterisk 12 ManagerEvent_MeetmeMute	407
6.3.62 Asterisk 12 ManagerEvent_MeetmeTalking	409
6.3.63 Asterisk 12 ManagerEvent_MeetmeTalkRequest	411
6.3.64 Asterisk 12 ManagerEvent_MessageWaiting	413
6.3.65 Asterisk 12 ManagerEvent_MiniVoiceMail	415
6.3.66 Asterisk 12 ManagerEvent_MonitorStart	416
6.3.67 Asterisk 12 ManagerEvent_MonitorStop	417
6.3.68 Asterisk 12 ManagerEvent_MusicOnHoldStart	418
6.3.69 Asterisk 12 ManagerEvent_MusicOnHoldStop	419
6.3.70 Asterisk 12 ManagerEvent_NewAccountCode	420
6.3.71 Asterisk 12 ManagerEvent_NewCallerid	421
6.3.72 Asterisk 12 ManagerEvent_Newchannel	422
6.3.73 Asterisk 12 ManagerEvent_NewExten	423
6.3.74 Asterisk 12 ManagerEvent_Newstate	424
6.3.75 Asterisk 12 ManagerEvent_OriginateResponse	425
6.3.76 Asterisk 12 ManagerEvent_ParkedCall	426
6.3.77 Asterisk 12 ManagerEvent_ParkedCallGiveUp	428
6.3.78 Asterisk 12 ManagerEvent_ParkedCallTimeOut	430
6.3.79 Asterisk 12 ManagerEvent_PeerStatus	432
6.3.80 Asterisk 12 ManagerEvent_Pickup	433
6.3.81 Asterisk 12 ManagerEvent_PresenceStatus	435
6.3.82 Asterisk 12 ManagerEvent_QueueCallerAbandon	436
6.3.83 Asterisk 12 ManagerEvent_QueueCallerJoin	437
6.3.84 Asterisk 12 ManagerEvent_QueueCallerLeave	439
6.3.85 Asterisk 12 ManagerEvent_QueueMemberAdded	440
6.3.86 Asterisk 12 ManagerEvent_QueueMemberPaused	441
6.3.87 Asterisk 12 ManagerEvent_QueueMemberPenalty	443
6.3.88 Asterisk 12 ManagerEvent_QueueMemberRemoved	444
6.3.89 Asterisk 12 ManagerEvent_QueueMemberRinginuse	445
6.3.90 Asterisk 12 ManagerEvent_QueueMemberStatus	446
6.3.91 Asterisk 12 ManagerEvent_ReceiveFAX	447
6.3.92 Asterisk 12 ManagerEvent_Registry	449
6.3.93 Asterisk 12 ManagerEvent_Reload	450
6.3.94 Asterisk 12 ManagerEvent_Rename	451
6.3.95 Asterisk 12 ManagerEvent_RTCPReceived	452
6.3.96 Asterisk 12 ManagerEvent_RTCPSent	454
6.3.97 Asterisk 12 ManagerEvent_SendFAX	456
6.3.98 Asterisk 12 ManagerEvent_SessionTimeout	458
6.3.99 Asterisk 12 ManagerEvent_Shutdown	459
6.3.100 Asterisk 12 ManagerEvent_SIPQualifyPeerDone	460
6.3.101 Asterisk 12 ManagerEvent_SoftHangupRequest	461
6.3.102 Asterisk 12 ManagerEvent_SpanAlarm	462
6.3.103 Asterisk 12 ManagerEvent_SpanAlarmClear	463
6.3.104 Asterisk 12 ManagerEvent_Status	464
6.3.105 Asterisk 12 ManagerEvent_Unhold	466
6.3.106 Asterisk 12 ManagerEvent_UnParkedCall	467
6.3.107 Asterisk 12 ManagerEvent_UserEvent	470
6.3.108 Asterisk 12 ManagerEvent_VarSet	471
6.4 Asterisk 12 ARI	474

6.4.1 Asterisk 12 REST Data Models	475
6.4.2 Asterisk 12 Asterisk REST API	507
6.4.3 Asterisk 12 Bridges REST API	508
6.4.4 Asterisk 12 Channels REST API	511
6.4.5 Asterisk 12 Endpoints REST API	518
6.4.6 Asterisk 12 Events REST API	519
6.4.7 Asterisk 12 Recordings REST API	520
6.4.8 Asterisk 12 Sounds REST API	523
6.4.9 Asterisk 12 Applications REST API	524
6.4.10 Asterisk 12 Playbacks REST API	526
6.4.11 Asterisk 12 Devicestates REST API	527
6.5 Asterisk 12 Dialplan Applications	528
6.5.1 Asterisk 12 Application_AddQueueMember	529
6.5.2 Asterisk 12 Application_ADSIProg	530
6.5.3 Asterisk 12 Application_AELSub	531
6.5.4 Asterisk 12 Application_AgentLogin	532
6.5.5 Asterisk 12 Application_AgentRequest	533
6.5.6 Asterisk 12 Application_AGI	534
6.5.7 Asterisk 12 Application_AlarmReceiver	535
6.5.8 Asterisk 12 Application_AMD	536
6.5.9 Asterisk 12 Application_Answer	537
6.5.10 Asterisk 12 Application_Authenticate	538
6.5.11 Asterisk 12 Application_BackGround	539
6.5.12 Asterisk 12 Application_BackgroundDetect	540
6.5.13 Asterisk 12 Application_Bridge	541
6.5.14 Asterisk 12 Application_BridgeWait	543
6.5.15 Asterisk 12 Application_Busy	544
6.5.16 Asterisk 12 Application_CallCompletionCancel	545
6.5.17 Asterisk 12 Application_CallCompletionRequest	546
6.5.18 Asterisk 12 Application_CELGenUserEvent	547
6.5.19 Asterisk 12 Application_ChangeMonitor	548
6.5.20 Asterisk 12 Application_ChansAvail	549
6.5.21 Asterisk 12 Application_ChannelRedirect	550
6.5.22 Asterisk 12 Application_ChanSpy	551
6.5.23 Asterisk 12 Application_ClearHash	553
6.5.24 Asterisk 12 Application_ConfBridge	554
6.5.25 Asterisk 12 Application_Congestion	555
6.5.26 Asterisk 12 Application_ContinueWhile	556
6.5.27 Asterisk 12 Application_ControlPlayback	557
6.5.28 Asterisk 12 Application_DAHDIAcceptR2Call	558
6.5.29 Asterisk 12 Application_DAHDIBarge	559
6.5.30 Asterisk 12 Application_DAHDIRAS	560
6.5.31 Asterisk 12 Application_DAHDIScan	561
6.5.32 Asterisk 12 Application_DAHDISendCallreroutingFacility	562
6.5.33 Asterisk 12 Application_DAHDISendKeypadFacility	563
6.5.34 Asterisk 12 Application_DateTime	564
6.5.35 Asterisk 12 Application_DBdel	565
6.5.36 Asterisk 12 Application_DBdeltree	566
6.5.37 Asterisk 12 Application_DeadAGI	567
6.5.38 Asterisk 12 Application_Dial	568
6.5.39 Asterisk 12 Application_Dictate	572
6.5.40 Asterisk 12 Application_Directory	573
6.5.41 Asterisk 12 Application_DISA	574
6.5.42 Asterisk 12 Application_DumpChan	575
6.5.43 Asterisk 12 Application_EAGI	576
6.5.44 Asterisk 12 Application_Echo	577
6.5.45 Asterisk 12 Application_EndWhile	578
6.5.46 Asterisk 12 Application_Exec	579
6.5.47 Asterisk 12 Application_ExecIf	580
6.5.48 Asterisk 12 Application_ExecIfTime	581
6.5.49 Asterisk 12 Application_ExitWhile	582

6.5.50 Asterisk 12 Application_ExtenSpy	583
6.5.51 Asterisk 12 Application_ExternalIVR	585
6.5.52 Asterisk 12 Application_Festival	586
6.5.53 Asterisk 12 Application_Flash	587
6.5.54 Asterisk 12 Application_FollowMe	588
6.5.55 Asterisk 12 Application_ForkCDR	589
6.5.56 Asterisk 12 Application_GetCPEID	590
6.5.57 Asterisk 12 Application_Gosub	591
6.5.58 Asterisk 12 Application_GosubIf	592
6.5.59 Asterisk 12 Application_Goto	593
6.5.60 Asterisk 12 Application_GotoIf	594
6.5.61 Asterisk 12 Application_GotoIfTime	595
6.5.62 Asterisk 12 Application_Hangup	596
6.5.63 Asterisk 12 Application_HangupCauseClear	597
6.5.64 Asterisk 12 Application_IAX2Provision	598
6.5.65 Asterisk 12 Application_ICES	599
6.5.66 Asterisk 12 Application_ImportVar	600
6.5.67 Asterisk 12 Application_Incomplete	601
6.5.68 Asterisk 12 Application_IVRDemo	602
6.5.69 Asterisk 12 Application_JabberJoin_res_jabber	603
6.5.70 Asterisk 12 Application_JabberJoin_res_xmpp	604
6.5.71 Asterisk 12 Application_JabberLeave_res_jabber	605
6.5.72 Asterisk 12 Application_JabberLeave_res_xmpp	606
6.5.73 Asterisk 12 Application_JabberSend_res_jabber	607
6.5.74 Asterisk 12 Application_JabberSend_res_xmpp	608
6.5.75 Asterisk 12 Application_JabberSendGroup_res_jabber	609
6.5.76 Asterisk 12 Application_JabberSendGroup_res_xmpp	610
6.5.77 Asterisk 12 Application_JabberStatus_res_jabber	611
6.5.78 Asterisk 12 Application_JabberStatus_res_xmpp	612
6.5.79 Asterisk 12 Application_JACK	613
6.5.80 Asterisk 12 Application_Log	614
6.5.81 Asterisk 12 Application_Macro	615
6.5.82 Asterisk 12 Application_MacroExclusive	616
6.5.83 Asterisk 12 Application_MacroExit	617
6.5.84 Asterisk 12 Application_MacroIf	618
6.5.85 Asterisk 12 Application_MailboxExists	619
6.5.86 Asterisk 12 Application_MeetMe	620
6.5.87 Asterisk 12 Application_MeetMeAdmin	622
6.5.88 Asterisk 12 Application_MeetMeChannelAdmin	623
6.5.89 Asterisk 12 Application_MeetMeCount	624
6.5.90 Asterisk 12 Application_MessageSend	625
6.5.91 Asterisk 12 Application_MilliWatt	626
6.5.92 Asterisk 12 Application_MinivmAccMess	627
6.5.93 Asterisk 12 Application_MinivmDelete	628
6.5.94 Asterisk 12 Application_MinivmGreet	629
6.5.95 Asterisk 12 Application_MinivmMWI	630
6.5.96 Asterisk 12 Application_MinivmNotify	631
6.5.97 Asterisk 12 Application_MinivmRecord	632
6.5.98 Asterisk 12 Application_MixMonitor	633
6.5.99 Asterisk 12 Application_Monitor	635
6.5.100 Asterisk 12 Application_Morsecode	636
6.5.101 Asterisk 12 Application_MP3Player	637
6.5.102 Asterisk 12 Application_MSet	638
6.5.103 Asterisk 12 Application_MusicOnHold	639
6.5.104 Asterisk 12 Application_NBScat	640
6.5.105 Asterisk 12 Application_NoCDR	641
6.5.106 Asterisk 12 Application_NoOp	642
6.5.107 Asterisk 12 Application_ODBC_Commit	643
6.5.108 Asterisk 12 Application_ODBC_Rollback	644
6.5.109 Asterisk 12 Application_ODBCFinish	645
6.5.110 Asterisk 12 Application_Originate	646

6.5.111 Asterisk 12 Application_OSPAuth	647
6.5.112 Asterisk 12 Application_OSPFinish	648
6.5.113 Asterisk 12 Application_OSPLookup	649
6.5.114 Asterisk 12 Application_OSPNext	651
6.5.115 Asterisk 12 Application_Page	652
6.5.116 Asterisk 12 Application_Park	653
6.5.117 Asterisk 12 Application_ParkAndAnnounce	654
6.5.118 Asterisk 12 Application_ParkedCall	655
6.5.119 Asterisk 12 Application_PauseMonitor	656
6.5.120 Asterisk 12 Application_PauseQueueMember	657
6.5.121 Asterisk 12 Application_Pickup	658
6.5.122 Asterisk 12 Application_PickupChan	659
6.5.123 Asterisk 12 Application_Playback	660
6.5.124 Asterisk 12 Application_PlayTones	661
6.5.125 Asterisk 12 Application_PrivacyManager	662
6.5.126 Asterisk 12 Application_Proceeding	663
6.5.127 Asterisk 12 Application_Progress	664
6.5.128 Asterisk 12 Application_Queue	665
6.5.129 Asterisk 12 Application_QueueLog	667
6.5.130 Asterisk 12 Application_RaiseException	668
6.5.131 Asterisk 12 Application_Read	669
6.5.132 Asterisk 12 Application_ReadExten	670
6.5.133 Asterisk 12 Application_ReadFile	671
6.5.134 Asterisk 12 Application_ReceiveFAX_app_fax	672
6.5.135 Asterisk 12 Application_ReceiveFAX_res_fax	673
6.5.136 Asterisk 12 Application_Record	674
6.5.137 Asterisk 12 Application_RemoveQueueMember	675
6.5.138 Asterisk 12 Application_ResetCDR	676
6.5.139 Asterisk 12 Application_RetryDial	677
6.5.140 Asterisk 12 Application_Return	678
6.5.141 Asterisk 12 Application_Ringing	679
6.5.142 Asterisk 12 Application_SayAlpha	680
6.5.143 Asterisk 12 Application_SayAlphaCase	681
6.5.144 Asterisk 12 Application_SayCountedAdj	682
6.5.145 Asterisk 12 Application_SayCountedNoun	683
6.5.146 Asterisk 12 Application_SayCountPL	684
6.5.147 Asterisk 12 Application_SayDigits	685
6.5.148 Asterisk 12 Application_SayNumber	686
6.5.149 Asterisk 12 Application_SayPhonetic	687
6.5.150 Asterisk 12 Application_SayUnixTime	688
6.5.151 Asterisk 12 Application_SendDTMF	689
6.5.152 Asterisk 12 Application_SendFAX_app_fax	690
6.5.153 Asterisk 12 Application_SendFAX_res_fax	691
6.5.154 Asterisk 12 Application_SendImage	692
6.5.155 Asterisk 12 Application_SendText	693
6.5.156 Asterisk 12 Application_SendURL	694
6.5.157 Asterisk 12 Application_Set	695
6.5.158 Asterisk 12 Application_SetAMAFIags	696
6.5.159 Asterisk 12 Application_SetCallerPres	697
6.5.160 Asterisk 12 Application_SetMusicOnHold	698
6.5.161 Asterisk 12 Application_SIPAddHeader	699
6.5.162 Asterisk 12 Application_SIPDtmfMode	700
6.5.163 Asterisk 12 Application_SIPRemoveHeader	701
6.5.164 Asterisk 12 Application_SIPSendCustomINFO	702
6.5.165 Asterisk 12 Application_SkelGuessNumber	703
6.5.166 Asterisk 12 Application_SLASStation	704
6.5.167 Asterisk 12 Application_SLATrunk	705
6.5.168 Asterisk 12 Application_SMS	706
6.5.169 Asterisk 12 Application_SoftHangup	707
6.5.170 Asterisk 12 Application_SpeechActivateGrammar	708
6.5.171 Asterisk 12 Application_SpeechBackground	709

6.5.172 Asterisk 12 Application_SpeechCreate	710
6.5.173 Asterisk 12 Application_SpeechDeactivateGrammar	711
6.5.174 Asterisk 12 Application_SpeechDestroy	712
6.5.175 Asterisk 12 Application_SpeechLoadGrammar	713
6.5.176 Asterisk 12 Application_SpeechProcessingSound	714
6.5.177 Asterisk 12 Application_SpeechStart	715
6.5.178 Asterisk 12 Application_SpeechUnloadGrammar	716
6.5.179 Asterisk 12 Application_StackPop	717
6.5.180 Asterisk 12 Application_StartMusicOnHold	718
6.5.181 Asterisk 12 Application_Stasis	719
6.5.182 Asterisk 12 Application_StopMixMonitor	720
6.5.183 Asterisk 12 Application_StopMonitor	721
6.5.184 Asterisk 12 Application_StopMusicOnHold	722
6.5.185 Asterisk 12 Application_StopPlayTones	723
6.5.186 Asterisk 12 Application_System	724
6.5.187 Asterisk 12 Application_TestClient	725
6.5.188 Asterisk 12 Application_TestServer	726
6.5.189 Asterisk 12 Application_Transfer	727
6.5.190 Asterisk 12 Application_TryExec	728
6.5.191 Asterisk 12 Application_TrySystem	729
6.5.192 Asterisk 12 Application_UnpauseMonitor	730
6.5.193 Asterisk 12 Application_UnpauseQueueMember	731
6.5.194 Asterisk 12 Application_UserEvent	732
6.5.195 Asterisk 12 Application_Verbose	733
6.5.196 Asterisk 12 Application_VMAuthenticate	734
6.5.197 Asterisk 12 Application_VMSayName	735
6.5.198 Asterisk 12 Application_VoiceMail	736
6.5.199 Asterisk 12 Application_VoiceMailMain	737
6.5.200 Asterisk 12 Application_VoiceMailPlayMsg	738
6.5.201 Asterisk 12 Application_Wait	739
6.5.202 Asterisk 12 Application_WaitExten	740
6.5.203 Asterisk 12 Application_WaitForNoise	741
6.5.204 Asterisk 12 Application_WaitForRing	742
6.5.205 Asterisk 12 Application_WaitForSilence	743
6.5.206 Asterisk 12 Application_WaitMusicOnHold	744
6.5.207 Asterisk 12 Application_WaitUntil	745
6.5.208 Asterisk 12 Application_While	746
6.5.209 Asterisk 12 Application_Zapateller	747
6.6 Asterisk 12 Dialplan Functions	748
6.6.1 Asterisk 12 Function_AES_DECRYPT	749
6.6.2 Asterisk 12 Function_AES_ENCRYPT	750
6.6.3 Asterisk 12 Function_AGC	751
6.6.4 Asterisk 12 Function_AGENT	752
6.6.5 Asterisk 12 Function_AMI_CLIENT	753
6.6.6 Asterisk 12 Function_ARRAY	754
6.6.7 Asterisk 12 Function_AST_CONFIG	755
6.6.8 Asterisk 12 Function_AUDIOHOOK_INHERIT	756
6.6.9 Asterisk 12 Function_BASE64_DECODE	757
6.6.10 Asterisk 12 Function_BASE64_ENCODE	758
6.6.11 Asterisk 12 Function_BLACKLIST	759
6.6.12 Asterisk 12 Function_CALENDAR_BUSY	760
6.6.13 Asterisk 12 Function_CALENDAR_EVENT	761
6.6.14 Asterisk 12 Function_CALENDAR_QUERY	762
6.6.15 Asterisk 12 Function_CALENDAR_QUERY_RESULT	763
6.6.16 Asterisk 12 Function_CALENDAR_WRITE	764
6.6.17 Asterisk 12 Function_CALLCOMPLETION	765
6.6.18 Asterisk 12 Function_CALLERID	766
6.6.19 Asterisk 12 Function_CALLERPRES	768
6.6.20 Asterisk 12 Function_CDR	769
6.6.21 Asterisk 12 Function_CDR_PROP	771
6.6.22 Asterisk 12 Function_CHANNEL	772

6.6.23 Asterisk 12 Function_CHANNELS	777
6.6.24 Asterisk 12 Function_CHECKSIPDOMAIN	778
6.6.25 Asterisk 12 Function_CONFBRIDGE	779
6.6.26 Asterisk 12 Function_CONFBRIDGE_INFO	780
6.6.27 Asterisk 12 Function_CONNECTEDLINE	781
6.6.28 Asterisk 12 Function_CSV_QUOTE	783
6.6.29 Asterisk 12 Function_CURL	784
6.6.30 Asterisk 12 Function_CURLOPT	785
6.6.31 Asterisk 12 Function_CUT	786
6.6.32 Asterisk 12 Function_DB	787
6.6.33 Asterisk 12 Function_DB_DELETE	788
6.6.34 Asterisk 12 Function_DB_EXISTS	789
6.6.35 Asterisk 12 Function_DB_KEYS	790
6.6.36 Asterisk 12 Function_DEC	791
6.6.37 Asterisk 12 Function_DENOISE	792
6.6.38 Asterisk 12 Function_DEVICE_STATE	793
6.6.39 Asterisk 12 Function_DIALGROUP	794
6.6.40 Asterisk 12 Function_DIALPLAN_EXISTS	795
6.6.41 Asterisk 12 Function_DUNDILOOKUP	796
6.6.42 Asterisk 12 Function_DUNDIQUERY	797
6.6.43 Asterisk 12 Function_DUNDIRESULT	798
6.6.44 Asterisk 12 Function_ENUMLOOKUP	799
6.6.45 Asterisk 12 Function_ENUMQUERY	800
6.6.46 Asterisk 12 Function_ENUMRESULT	801
6.6.47 Asterisk 12 Function_ENV	802
6.6.48 Asterisk 12 Function_EVAL	803
6.6.49 Asterisk 12 Function_EXCEPTION	804
6.6.50 Asterisk 12 Function_EXISTS	805
6.6.51 Asterisk 12 Function_EXTENSION_STATE	806
6.6.52 Asterisk 12 Function_FAXOPT_res_fax	807
6.6.53 Asterisk 12 Function_FEATURE	808
6.6.54 Asterisk 12 Function_FEATUREMAP	809
6.6.55 Asterisk 12 Function_FIELDNUM	810
6.6.56 Asterisk 12 Function_FIELDQTY	811
6.6.57 Asterisk 12 Function_FILE	812
6.6.58 Asterisk 12 Function_FILE_COUNT_LINE	814
6.6.59 Asterisk 12 Function_FILE_FORMAT	815
6.6.60 Asterisk 12 Function_FILTER	816
6.6.61 Asterisk 12 Function_FRAME_TRACE	817
6.6.62 Asterisk 12 Function_GLOBAL	818
6.6.63 Asterisk 12 Function_GROUP	819
6.6.64 Asterisk 12 Function_GROUP_COUNT	820
6.6.65 Asterisk 12 Function_GROUP_LIST	821
6.6.66 Asterisk 12 Function_GROUP_MATCH_COUNT	822
6.6.67 Asterisk 12 Function_HANGUPCAUSE	823
6.6.68 Asterisk 12 Function_HANGUPCAUSE_KEYS	824
6.6.69 Asterisk 12 Function_HASH	825
6.6.70 Asterisk 12 Function_HASHKEYS	826
6.6.71 Asterisk 12 Function_HINT	827
6.6.72 Asterisk 12 Function_IAXPEER	828
6.6.73 Asterisk 12 Function_IAXVAR	829
6.6.74 Asterisk 12 Function_ICONV	830
6.6.75 Asterisk 12 Function_IF	831
6.6.76 Asterisk 12 Function_IFMODULE	832
6.6.77 Asterisk 12 Function_IFTIME	833
6.6.78 Asterisk 12 Function_IMPORT	834
6.6.79 Asterisk 12 Function_INC	835
6.6.80 Asterisk 12 Function_ISNULL	836
6.6.81 Asterisk 12 Function_JABBER_RECEIVE_res_jabber	837
6.6.82 Asterisk 12 Function_JABBER_RECEIVE_res_xmpp	838
6.6.83 Asterisk 12 Function_JABBER_STATUS_res_jabber	839

6.6.84 Asterisk 12 Function_JABBER_STATUS_res_xmpp	840
6.6.85 Asterisk 12 Function_JITTERBUFFER	841
6.6.86 Asterisk 12 Function_KEYPADHASH	842
6.6.87 Asterisk 12 Function_LEN	843
6.6.88 Asterisk 12 Function_LISTFILTER	844
6.6.89 Asterisk 12 Function_LOCAL	845
6.6.90 Asterisk 12 Function_LOCAL_PEEK	846
6.6.91 Asterisk 12 Function_LOCK	847
6.6.92 Asterisk 12 Function_MAILBOX_EXISTS	848
6.6.93 Asterisk 12 Function_MASTER_CHANNEL	849
6.6.94 Asterisk 12 Function_MATH	850
6.6.95 Asterisk 12 Function_MD5	851
6.6.96 Asterisk 12 Function_MEETME_INFO	852
6.6.97 Asterisk 12 Function_MESSAGE	853
6.6.98 Asterisk 12 Function_MESSAGE_DATA	854
6.6.99 Asterisk 12 Function_MINIVMACCOUNT	855
6.6.100 Asterisk 12 Function_MINIVMCOUNTER	856
6.6.101 Asterisk 12 Function_MUTEAUDIO	857
6.6.102 Asterisk 12 Function_ODBC	858
6.6.103 Asterisk 12 Function_ODBC_FETCH	859
6.6.104 Asterisk 12 Function_PASSTHRU	860
6.6.105 Asterisk 12 Function_PITCH_SHIFT	861
6.6.106 Asterisk 12 Function_PJSIP_DIAL_CONTACTS	862
6.6.107 Asterisk 12 Function_PJSIP_ENDPOINT	863
6.6.108 Asterisk 12 Function_PJSIP_HEADER	865
6.6.109 Asterisk 12 Function_PJSIP_MEDIA_OFFER	867
6.6.110 Asterisk 12 Function_POP	868
6.6.111 Asterisk 12 Function_PP_EACH_EXTENSION	869
6.6.112 Asterisk 12 Function_PP_EACH_USER	870
6.6.113 Asterisk 12 Function_PRESENCE_STATE	871
6.6.114 Asterisk 12 Function_PUSH	872
6.6.115 Asterisk 12 Function_QUEUE_EXISTS	873
6.6.116 Asterisk 12 Function_QUEUE_MEMBER	874
6.6.117 Asterisk 12 Function_QUEUE_MEMBER_COUNT	875
6.6.118 Asterisk 12 Function_QUEUE_MEMBER_LIST	876
6.6.119 Asterisk 12 Function_QUEUE_MEMBER_PENALTY	877
6.6.120 Asterisk 12 Function_QUEUE_VARIABLES	878
6.6.121 Asterisk 12 Function_QUEUE_WAITING_COUNT	879
6.6.122 Asterisk 12 Function_QUOTE	880
6.6.123 Asterisk 12 Function_RAND	881
6.6.124 Asterisk 12 Function_REALTIME	882
6.6.125 Asterisk 12 Function_REALTIME_DESTROY	883
6.6.126 Asterisk 12 Function_REALTIME_FIELD	884
6.6.127 Asterisk 12 Function_REALTIME_HASH	885
6.6.128 Asterisk 12 Function_REALTIME_STORE	886
6.6.129 Asterisk 12 Function_REDIRECTING	887
6.6.130 Asterisk 12 Function_REGEX	890
6.6.131 Asterisk 12 Function_REPLACE	891
6.6.132 Asterisk 12 Function_SET	892
6.6.133 Asterisk 12 Function_SHA1	893
6.6.134 Asterisk 12 Function_SHARED	894
6.6.135 Asterisk 12 Function_SHELL	895
6.6.136 Asterisk 12 Function_SHIFT	896
6.6.137 Asterisk 12 Function_SIP_HEADER	897
6.6.138 Asterisk 12 Function_SIPCHANINFO	898
6.6.139 Asterisk 12 Function_SIPPEER	899
6.6.140 Asterisk 12 Function_SMDI_MSG	900
6.6.141 Asterisk 12 Function_SMDI_MSG_RETRIEVE	901
6.6.142 Asterisk 12 Function_SORT	902
6.6.143 Asterisk 12 Function_SPEECH	903
6.6.144 Asterisk 12 Function_SPEECH_ENGINE	904

6.6.145 Asterisk 12 Function_SPEECH_GRAMMAR	905
6.6.146 Asterisk 12 Function_SPEECH_RESULTS_TYPE	906
6.6.147 Asterisk 12 Function_SPEECH_SCORE	907
6.6.148 Asterisk 12 Function_SPEECH_TEXT	908
6.6.149 Asterisk 12 Function_SPRINTF	909
6.6.150 Asterisk 12 Function_SQL_ESC	910
6.6.151 Asterisk 12 Function_SRVQUERY	911
6.6.152 Asterisk 12 Function_SRVRESULT	912
6.6.153 Asterisk 12 Function_STACK_PEEK	913
6.6.154 Asterisk 12 Function_STAT	914
6.6.155 Asterisk 12 Function_STRFTIME	915
6.6.156 Asterisk 12 Function_STRPTIME	916
6.6.157 Asterisk 12 Function_STRREPLACE	917
6.6.158 Asterisk 12 Function_SYSINFO	918
6.6.159 Asterisk 12 Function_TESTTIME	920
6.6.160 Asterisk 12 Function_TIMEOUT	921
6.6.161 Asterisk 12 Function_TOLOWER	922
6.6.162 Asterisk 12 Function_TOUPPER	923
6.6.163 Asterisk 12 Function_TRYLOCK	924
6.6.164 Asterisk 12 Function_TXTCIDNAME	925
6.6.165 Asterisk 12 Function_UNLOCK	926
6.6.166 Asterisk 12 Function_UNSHIFT	927
6.6.167 Asterisk 12 Function_URIDECODE	928
6.6.168 Asterisk 12 Function_URIENCODE	929
6.6.169 Asterisk 12 Function_VALID_EXTEN	930
6.6.170 Asterisk 12 Function_VERSION	931
6.6.171 Asterisk 12 Function_VM_INFO	932
6.6.172 Asterisk 12 Function_VMCOUNT	933
6.6.173 Asterisk 12 Function_VOLUME	934
6.7 Asterisk 12 Module Configuration	935
6.7.1 Asterisk 12 Configuration_app_agent_pool	936
6.7.2 Asterisk 12 Configuration_app_confbridge	939
6.7.3 Asterisk 12 Configuration_app_skel	946
6.7.4 Asterisk 12 Configuration_ari	948
6.7.5 Asterisk 12 Configuration_cdr	950
6.7.6 Asterisk 12 Configuration_cel	953
6.7.7 Asterisk 12 Configuration_chan_motif	954
6.7.8 Asterisk 12 Configuration_core	956
6.7.9 Asterisk 12 Configuration_features	957
6.7.10 Asterisk 12 Configuration_named_acl	961
6.7.11 Asterisk 12 Configuration_res_ari	962
6.7.12 Asterisk 12 Configuration_res_parking	964
6.7.13 Asterisk 12 Configuration_res_pjsip	968
6.7.14 Asterisk 12 Configuration_res_pjsip_acl	984
6.7.15 Asterisk 12 Configuration_res_pjsip_endpoint_identifier_ip	986
6.7.16 Asterisk 12 Configuration_res_pjsip_notify	987
6.7.17 Asterisk 12 Configuration_res_pjsip_outbound_registration	988
6.7.18 Asterisk 12 Configuration_res_statsd	990
6.7.19 Asterisk 12 Configuration_res_xmpp	991
6.7.20 Asterisk 12 Configuration_stasis	994
6.7.21 Asterisk 12 Configuration_udptl	995

New in 12

Overview

Asterisk 12 is a standard release of the Asterisk project. As such, the focus of development for this release was on core architectural changes and major new features. This includes:

- A more flexible bridging core based on the Bridging API
- A new internal message bus, Stasis
- Major standardization and consistency improvements to AMI
- Addition of the Asterisk REST Interface (ARI)
- A new SIP channel driver, chan_pjsip

In addition, as the vast majority of bridging in Asterisk was migrated to the Bridging API used by ConfBridge, major changes were made to most of the interfaces in Asterisk. This includes not only AMI, but also CDRs and CEL.

Specifications have been written for the affected interfaces:

- [AMI v2 Specification](#)
- [Asterisk 12 CEL Specification](#)
- [Asterisk 12 CDR Specification](#)

It is **highly** recommended that anyone migrating to Asterisk 12 read the information regarding its release both in the [CHANGES](#) files and in the accompanying [UPGRADE.txt](#) file.

Build System

- Added build option `DISABLE_INLINE`. This option can be used to work around a bug in gcc. For more information, see http://gcc.gnu.org/bugzilla/show_bug.cgi?id=47816
- Removed the `CHANNEL_TRACE` development mode build option. Certain aspects of the `CHANNEL_TRACE` build option were incompatible with the new bridging architecture.
- Asterisk now optionally uses `libxslt` to improve XML documentation generation and maintainability. If `libxslt` is not available on the system, some XML documentation will be incomplete.
- Asterisk now depends on `libjansson`. If a package of `libjansson` is not available on your distro, please see <http://www.digip.org/jansson/>.
- Asterisk now depends on `libuuid` and, optionally, `uriparser`. It is recommended that you install `uriparser`, even if it is optional.
- The new SIP stack and channel driver currently use a particular version of PJSIP. Please see <https://wiki.asterisk.org/wiki/x/J4GLAQ> for more information on configuring and installing PJSIP for usage with Asterisk.

Applications

AgentLogin

- Along with [AgentRequest](#), this application has been modified to be a replacement for `chan_agent`. The act of a channel calling the AgentLogin application places the channel into a pool of agents that can be requested by the AgentRequest application. Note that this application, as well as all other agent related functionality, is now provided by the `app_agent_pool` module. See [chan_agent](#) and [AgentRequest](#) for more information.
- This application no longer performs agent authentication. If authentication is desired, the dialplan needs to perform this function using the [Authenticate](#) or [VMAuthenticate](#) application or through an AGI script before running AgentLogin.
- If this application is called and the agent is already logged in, the dialplan will continue execution with the `AGENT_STATUS` channel variable set to `ALREADY_LOGGED_IN`.
- The [agents.conf](#) schema has changed. Rather than specifying agents on a single line in comma delineated fashion, each agent is defined in a separate context. This allows agents to use the power of context templates in their definition.
- A number of parameters from [agents.conf](#) have been removed. This includes:
 - `maxloginretries`
 - `autologoffunavail`

- `updatecdr`
- `goodbye`
- `group`
- `recordformat`
- `urlprefix`
- `savecallsin`

These options were obsoleted by the move from a channel driver model to the bridging/application model provided by `app_agent_pool`.

AgentRequest

- A new application, this will request a logged in agent from the pool and bridge the requested channel with the channel calling this application. Logged in agents are those channels that called the [AgentLogin](#) application. If an agent cannot be requested from the pool, the `AGENT_STATUS` dialplan application will be set with an appropriate error value.

AgentMonitorOutgoing

- This application has been removed. It was a holdover from when `AgentCallbackLogin` was removed in Asterisk 1.6.0.

AlarmReceiver

- **Added support for additional Ademco DTMF signalling formats, including Express 4+1, Express 4+2, High Speed and Super Fast.**
- Added channel variable `ALARMRECEIVER_CALL_LIMIT`. This sets the maximum call time, in milliseconds, to run the application.
- Added channel variable `ALARMRECEIVER_RETRIES_LIMIT`. This sets the maximum number of times to retry the call.
- Added a new configuration option `answait`. If set, the `AlarmReceiver` application will wait the number of milliseconds specified by `answait` after the channel has answered. Valid values range between 500 milliseconds and 10000 milliseconds.
- Added configuration option `no_group_meta`. If enabled, grouping of metadata information in the `AlarmReceiver` log file will be skipped.

Answer

- It is now no longer possible to bypass updating the CDR on the channel when answering. CDRs reflect the state of the channel and will always reflect the time they were Answered.

BridgeWait

- A new application in Asterisk, this will place the calling channel into a holding bridge, optionally entertaining them with some form of media. Channels participating in a holding bridge do not interact with other channels in the same holding bridge. Optionally, however, a channel may join as an announcer. Any media passed from an announcer channel is played to all channels in the holding bridge. Channels leave a holding bridge either when an optional timer expires, or via the [ChannelRedirect](#) application or [AMI Redirect](#) action.

ConfBridge

- All participants in a bridge can now be kicked out of a conference room by specifying the channel parameter as 'all' in the `ConfBridge` kick CLI command, i.e., `confbridge kick <conference> all`
- CLI output for the `confbridge list` command has been improved. When displaying information about a particular bridge, flags will now be shown for the participating users indicating properties of that user.
- The `ConfbridgeList` event now contains the following fields: `WaitMarked`, `EndMarked`, and `Waiting`. This displays additional properties about the user's profile, as well as whether or not the user is waiting for a Marked user to enter the conference.
- Added a new option for conference recording, `record_file_append`. If enabled, when the recording is stopped and then re-started, the existing recording will be used and appended to.
`ConfBridge` now has the ability to set the language of announcements to the conference. The language can be set on a bridge profile in `confbridge.conf` or by the dialplan function `CONFBRIDGE(bridge, language)={language}`.

ControlPlayback

- The channel variable `CPLAYBACKSTATUS` may now return the value `REMOTESTOPPED`. This occurs when playback is stopped by a remote interface, such as AMI. See the AMI action [ControlPlayback](#) for more information.

Directory

- Added the `a` option, which allows the caller to enter in an additional alias for the user in the directory. This option must be used in conjunction with the `f`, `l`, or `b` options. Note that the alias for a user can be specified in *voicemail.conf*.

DumpChan

- The output of `DumpChan` no longer includes the `DirectBridge` or `IndirectBridge` fields. Instead, if a channel is in a bridge, it includes a `BridgeID` field containing the unique ID of the bridge that the channel happens to be in.

ForkCDR

- `ForkCDR` no longer automatically resets the forked CDR. See the `r` option for more information.
- Variables are no longer purged from the original CDR. See the `v` option for more information.
- The `A` option has been removed. The Answer time on a CDR is never updated once set.
- The `d` option has been removed. The disposition on a CDR is a function of the state of the channel and cannot be altered.
- The `D` option has been removed. Who the Party B is on a CDR is a function of the state of the respective channels involved in the CDR and cannot be altered.
- The `r` option has been changed. Previously, `ForkCDR` always reset the CDR such that the start time and, if applicable, the answer time was updated. Now, by default, `ForkCDR` simply forks the CDR, maintaining any times. The `r` option now triggers the Reset, setting the start time (and answer time if applicable) to the current time. Note that the `a` option still sets the answer time to the current time if the channel was already answered.
- The `s` option has been removed. A variable can be set on the original CDR if desired using the `CDR` function, and removed from a forked CDR using the same function.
- The `T` option has been removed. The concept of `DONT_TOUCH` and `LOCKED` no longer applies in the CDR engine.
- The `v` option now prevents the copy of the variables from the original CDR to the forked CDR. Previously the variables were always copied but were removed from the original. This was changed as removing variables from a CDR can have unintended side effects - this option allows the user to prevent propagation of variables from the original to the forked without modifying the original.

MeetMe

- Added the `n` option to `MeetMe` to prevent application of the [DENOISE](#) function to a channel joining a conference. Some channel drivers that vary the number of audio samples in a voice frame will experience significant quality problems if a denoiser is attached to the channel; this option gives them the ability to remove the denoiser without having to unload `func_speex`.

MixMonitor

- The `b` option now includes conferences as well as sounds played to the participants.
- The [AUDIOHOOK_INHERIT](#) function is no longer needed to keep a `MixMonitor` running during a transfer. If a `MixMonitor` is started on a channel, the `MixMonitor` will continue to record the audio passing through the channel even in the presence of transfers.

NoCDR

- The `NoCDR` application is deprecated. Please use the [CDR_PROP](#) function to disable CDRs.
- While the `NoCDR` application will prevent CDRs for a channel from being propagated to registered CDR backends, it will not prevent that data from being collected. Hence, a subsequent call to [ResetCDR](#) or the [CDR_PROP](#) function that enables CDRs on a channel will restore those records that have not yet been finalized.

ParkAndAnnounce

- The `app_parkandannounce` module has been removed. The application ParkAndAnnounce is now provided by the `res_parking` module. See the [res_parking](#) changes for more information.

Queue

- Added queue available hint. The hint can be added to the dialplan using the following syntax:

```
exten => {exten},hint,Queue:{queue_name}_avail
```

For example, if the name of the queue is `markq` and the extension is 8501:

```
exten => 8501,hint,Queue:markq_avail
```

This will report `In Use` if there are no logged in agents or no free agents. It will report `Idle` when an agent is free.

- Queues now support a hint for member paused state. The hint uses the following syntax:

```
exten => {exten},hint,Queue:{queue_name}_pause_{member_name}
```

Where `queue_name` and `member_name` are the name of the queue and the name of the member to subscribe to, respectively. For example, for the sales queue, with queue member `mark` at extension 8501:

```
exten => 8501,hint,Queue:sales_pause_mark
```

Members will show as `In Use` when paused.

- The configuration options `eventwhencalled` and `eventmemberstatus` have been removed. As a result, the AMI events [QueueMemberStatus](#), [AgentCalled](#), [AgentConnect](#), [AgentComplete](#), [AgentDump](#), and [AgentRingNoAnswer](#) will always be sent. The *Variable* fields will also no longer exist on the `Agent*` events. These events can be filtered out from a connected AMI client using the `eventfilter` setting in *manager.conf*.
- The queue log now differentiates between blind and attended transfers. A blind transfer will result in a `BLINDTRANSFER` message with the destination context and extension. An attended transfer will result in an `ATTENDEDTRANSFER` message. This message will indicate the method by which the attended transfer was completed: `BRIDGE` for a bridge merge, `APP` for running an application on a bridge or channel, or `LINK` for linking two bridges together with local channels. The queue log will also now detect externally initiated blind and attended transfers and record the transfer status accordingly.
- When performing queue pause/unpause on an interface without specifying an individual queue, the `PAUSEALL/UNPAUSEALL` event will only be logged if at least one member of any queue exists for that interface.
- Added the `queue_log_realtime_use_gmt` option to have timestamps in GMT for realtime queue log entries.

ResetCDR

- The `e` option has been deprecated. Use the [CDR_PROP](#) function to re-enable CDRs when they were previously disabled on a channel.
- The `w` and `a` options have been removed. Dispatching CDRs to registered backends occurs on an as-needed basis in order to preserve `linkedid` propagation and other needed behavior.

SayAlphaCase

- A new application, this is similar to [SayAlpha](#) except that it supports case sensitive playback of the specified characters. For example:

```
same => n,SayAlphaCase(u,aBc)
```

Will result in 'a uppercase b c'.

SetAMAFlags

- This application is deprecated in favor of [CHANNEL\(amaflags\)](#).

SendDTMF

- The SendDTMF application will now accept `w` as valid input. This will cause the application to delay one second while streaming DTMF.

Stasis

- A new application in Asterisk 12, this hands control of the channel calling the application over to an external system. Currently, external systems manipulate channels in Stasis through the [Asterisk REST Interface \(ARI\)](#).

UserEvent

- UserEvent will now handle duplicate keys by overwriting the previous value assigned to the key.
- In addition to AMI, UserEvent invocations will now be distributed to any interested [Stasis](#) applications.

VoiceMail

- The *voicemail.conf* configuration file now has an *alias* configuration parameter for use with the [Directory](#) application. The voicemail realtime database table schema has also been updated with an *alias* column.
- Mailboxes defined in *voicemail.conf* **MUST** be referenced by their full name by entities that want to subscribe for MWI, i.e., `mailbox@context`. Previously, Asterisk would automatically append "@default" to a mailbox name if it wasn't specified; however, in order to support other providers of voicemail, this is no longer possible.

Codecs

- Pass through support has been added for both VP8 and Opus.
- Added format attribute negotiation for the Opus codec. Format attribute negotiation is provided by the `res_format_attr_opus` module.

Core

- Masquerades as an operation inside Asterisk have been effectively hidden by the migration to the Bridging API. As such, many 'quirks' of Asterisk no longer occur. This includes renaming of channels, "<ZOMBIE>" channels, dropping of frame/audio hooks, and other internal implementation details that users had to deal with. This fundamental change has large implications throughout the changes documented for this version.
- Multiple parties in a bridge may now be transferred. If a participant in a multi-party bridge initiates a blind transfer, a Local channel will be used to execute the dialplan location that the transferer sent the parties to. If a participant in a multi-party bridge initiates an attended transfer, several options are possible. If the attended transfer results in a transfer to an application, a Local channel is used. If the attended transfer results in a transfer to another channel, the resulting channels will be merged into a single bridge.
- The channel variable `ATTENDED_TRANSFER_COMPLETE_SOUND` is no longer channel driver specific. If the channel variable is set on the transferer channel, the sound will be played to the target of an attended transfer.
- The channel variable `BRIDGEPEER` becomes a comma separated list of peers in a multi-party bridge. The `BRIDGEPEER` value can have a maximum of 10 peers listed. Any more peers in the bridge will not be included in the list. `{{BRIDGEPEER}}` is not valid in holding bridges like parking since those channels do not talk to each other even though they are in a bridge.
- The channel variable `BRIDGEPVTCALLID` is only valid for two party bridges and will contain a value if the `BRIDGEPEER`'s channel driver supports it.
- A channel variable `ATTENDEDTRANSFER` is now set which indicates which channel was responsible for an attended transfer in a similar fashion to `BLINDTRANSFER`.
- Modules using the Configuration Framework or Sorcery must have XML configuration documentation. This configuration documentation is included with the rest of Asterisk's XML documentation, and is accessible via CLI commands. See the [CLI changes](#) for more information.

AMI (Asterisk Manager Interface)

- Major changes were made to both the syntax as well as the semantics of the AMI protocol. In particular, AMI events have been substantially improved in this version of Asterisk. For more information, please see the [AMI specification](#).
 - [AMI events that reference a particular channel or bridge](#) will now always contain a standard set of fields. When multiple channels or bridges are referenced in an event, fields for at least some subset of the channels and bridges in the event will be prefixed with a descriptive name to avoid name collisions. See the [AMI event documentation](#) for more information.
 - The CLI command `manager show commands` no longer truncates command names longer than 15 characters and no longer shows authorization requirement for commands. `manager show command` now displays the privileges needed for using a given manager command instead.
 - The [SIPshowpeer](#) action will now include a *SubscribeContext* field for a peer in its response if the peer has a subscribe context set.
 - The [SIPqualifypeer](#) action now acknowledges the request once it has established that the request is against a known peer. It also issues a new event, [SIPQualifyPeerDone](#), once the qualify action has been completed.
 - The [PlayDTMF](#) action now supports an optional *Duration* parameter. This specifies the duration of the digit to be played, in milliseconds.
 - Added [VoicemailRefresh](#) action to allow an external entity to trigger mailbox updates when changes occur instead of requiring the use of `pollmailboxes`.
-
- Added a new action [ControlPlayback](#). The [ControlPlayback](#) action allows an AMI client to manipulate audio currently being played back on a channel. The supported operations depend on the application being used to send audio to the channel. When the audio playback was initiated using the [ControlPlayback](#) application or [CONTROL STREAM FILE](#) AGI command, the audio can be paused, stopped, restarted, reversed, or skipped forward. When initiated by other mechanisms (such as the [Playback](#) application), the audio can be stopped, reversed, or skipped forward.
 - Channel related events now contain a snapshot of channel state, adding new fields to many of these events.
 - The AMI event [Newexten](#) field *Extension* is deprecated, and may be removed in a future release. Please use the common *Exten* field instead.
 - The AMI event [UserEvent](#) from `app_userevent` now contains the channel state fields. The channel state fields will come before the body fields.
 - The AMI events [ParkedCall](#), [ParkedCallTimeOut](#), [ParkedCallGiveUp](#), and [UnParkedCall](#) have changed significantly in the new `res_parking` module.
 - The *Channel* and *From* headers are gone.
 - For the channel that was parked or is coming out of parking, a *Parkee* channel snapshot is issued and it has a number of fields associated with it. The old *Channel* header relayed the same data as the new *ParkeeChannel* header.
 - The *From* field was ambiguous and changed meaning depending on the event. For most of these, it was the name of the channel that parked the call (the *Parker*).
 - There is no longer a header that provides this channel name, however the *ParkerDialString* will contain a dialstring to redial the device that parked the call.
 - On [UnParkedCall](#) events, the *From* header would instead represent the channel responsible for retrieving the parkee. It receives a channel snapshot labeled *Retriever*. The *From* field is replaced with *RetrieverChannel*.
 - Lastly, the *Exten* field has been replaced with *ParkingSpace*.
 - The AMI event [Parkinglot](#) (response to [Parkinglots](#) command) in a similar fashion has changed the field names *StartExten* and *StopExten* to *StartSpace* and *StopSpace* respectively.
 - The deprecated use of `|` (pipe) as a separator in the `channelvars` setting in `manager.conf` has been removed.
 - Channel Variables conveyed with a channel no longer contain the name of the channel as part of the key field, i.e., `ChanVariable(SIP/foo): bar=baz` is now `ChanVariable: bar=baz`. When multiple channels are present in a single AMI event, the various *ChanVariable* fields will contain a prefix that specifies which channel they correspond to.
 - The [NewPeerAccount](#) AMI event is no longer raised. The [NewAccountCode](#) AMI event always conveys the AMI event for a particular channel.
 - All [Reload](#) events have been consolidated into a single event type. This event will always contain a *Module* field specifying the name of the module and a *Status* field denoting the result of the reload. All modules now issue this event when being reloaded.
 - The [ModuleLoadReport](#) event has been removed. Most AMI connections would fail to receive this event due to being connected after modules have loaded. AMI connections that want to know when Asterisk is ready should listen for the [FullyBooted](#) event.
 - `app_fax` now sends the same send fax/receive fax events as `res_fax`. The [FaxSent](#) event is now the [SendFAX](#) event, and the [FaxReceived](#) event is now the [ReceiveFAX](#) event.
 - The [MusicOnHold](#) event is now two events: [MusicOnHoldStart](#) and [MusicOnHoldStop](#). The sub type field has been removed.
 - The [JabberEvent](#) event has been removed. It is not AMI's purpose to be a carrier for another protocol.
 - The [Bridge](#) Manager action's *Playtone* header now accepts more fine-grained options. *Channel1* and *Channel2* may be specified in order to play a tone to the specific channel. *Both* may be specified to play a tone to both channels. The old *yes* option is still accepted as a way of playing the tone to *Channel2* only.
 - The AMI [Status](#) response event to the AMI [Status](#) action replaces the *BridgedChannel* and *BridgedUniqueid* headers with the *BridgeID* header to indicate what bridge the channel is currently in.
 - The AMI [Hold](#) event has been moved out of individual channel drivers, into core, and is now two events: [Hold](#) and [Unhold](#). The status field has been removed.

- The AMI events in `app_queue` have been made more consistent with each other. Events that reference channels (`QueueCaller*` and `Agent*`) will show information about each channel. The (infamous) Join and Leave AMI events have been changed to [QueueCallerJoin](#) and [QueueCallerLeave](#).
- The `MCID` AMI event now publishes a channel snapshot when available and its non-channel-snapshot parameters now use either the `MCallerID` or `MConnectedID` prefixes with `Subaddr*`, `Name*`, and `Num*` suffixes instead of `CallerID` and `ConnectedID` to avoid confusion with similarly named parameters in the channel snapshot.
- The AMI events `Agentlogin` and `Agentlogoff` have been renamed [AgentLogin](#) and [AgentLogoff](#) respectively.
- The `Channel` key used in the [AlarmClear](#), [Alarm](#), and [DNDState](#) has been renamed [DAHDIChannel](#) since it does not convey an Asterisk channel name.
- `ChannelUpdate` events have been removed.
- All AMI events now contain a `SystemName` field, if available.
- Local channel optimization is now conveyed in two events: [LocalOptimizationBegin](#) and [LocalOptimizationEnd](#). The Begin event is sent when the Local channel driver begins attempting to optimize itself out of the media path; the End event is sent after the channel halves have successfully optimized themselves out of the media path.
- Local channel information in events is now prefixed with `LocalOne` and `LocalTwo`. This replaces the suffix of '1' and '2' for the two halves of the Local channel. This affects the [LocalBridge](#), [LocalOptimizationBegin](#), and [LocalOptimizationEnd](#) events.
- The option `allowmultiplelogin` can now be set or overridden in a particular account. When set in the general context, it will act as the default setting for defined accounts.
- The `BridgeAction` event was removed. It technically added no value, as the [Bridge](#) Action already receives confirmation of the bridge through a successful completion Event.
- The `BridgeExec` events were removed. These events duplicated the events that occur in the Bridging API, and are conveyed now through [BridgeCreate](#), [BridgeEnter](#), and [BridgeLeave](#) events.
- The [RTCPSent/RTCPReceived](#) events have been significantly modified from previous versions. They now report all SR/RR packets sent/received, and have been restructured to better reflect the data sent in a SR/RR. In particular, the event structure now supports multiple report blocks.
- Added [BlindTransfer](#) and [AttendedTransfer](#) events. These events are raised when a blind transfer/attended transfer completes successfully. They contain information about the transfer that just completed, including the location of the transferred channel.
- Added a 'security' class authorization for security events emitted by AMI. Note that these events are produced by the Asterisk Security Event framework.

CDR (Call Detail Records)

- Significant changes have been made to the behavior of CDRs. The CDR engine was effectively rewritten and built on the Stasis message bus. For a full definition of CDR behavior in Asterisk 12, please read the [CDR specification for Asterisk 12](#).
- CDRs will now be created between all participants in a bridge. For each pair of channels in a bridge, a CDR is created to represent the path of communication between those two endpoints. This lets an end user choose who to bill for what during bridge operations with multiple parties.
- The `duration`, `billsec`, `start`, `answer`, and `end` times now reflect the times associated with the current CDR for the channel, as opposed to a cumulative measurement of all CDRs for that channel.
- When a CDR is dispatched, user defined CDR variables from both parties are included in the resulting CDR. If both parties have the same variable, only the Party A value is provided.
- Added a new option to `cdr.conf`, `debug`. When enabled, significantly more information regarding the CDR engine is logged as verbose messages. This option should only be used if the behavior of the CDR engine needs to be debugged.
- Added CLI command `cdr set debug {on|off}`. This toggles the `debug` setting normally configured in `cdr.conf`.
- Added CLI command `cdr show active {channel}`. When `{channel}` is not specified, this command provides a summary of the channels with CDR information and their statistics. When `{channel}` is specified, it shows detailed information about all records associated with `{channel}`.

CEL (Channel Event Logging)

- CEL has undergone significant rework in Asterisk 12, and is now built on the Stasis message bus. Please see the [specification for CEL](#) for more detailed information.
- The `extra` field of all CEL events that use it now consists of a JSON blob with key/value pairs which are defined in the Asterisk 12 CEL documentation. This JSON blob will be interpreted by the various CEL backends into their specific format.
- `BLINDTRANSFER` events now report the transferee bridge unique identifier, extension, and context in a JSON blob as the extra string instead of the transferee channel name as the peer.
- `ATTENDEDTRANSFER` events now report the peer as NULL and additional information in the 'extra' string as a JSON blob. For transfers that occur between two bridged channels, the 'extra' JSON blob contains the primary bridge unique identifier, the secondary channel name, and the secondary bridge unique identifier. For transfers that occur between a bridged channel and a channel running an app, the

'extra' JSON blob contains the primary bridge unique identifier, the secondary channel name, and the app name.

- `LOCAL_OPTIMIZE` events have been added to convey local channel optimizations with the record occurring for the semi-one channel and the semi-two channel name in the peer field.
- `BRIDGE_START`, `BRIDGE_END`, `BRIDGE_UPDATE`, `3WAY_START`, `3WAY_END`, `CONF_ENTER`, `CONF_EXIT`, `CONF_START`, and `CONF_END` events have all been removed. These events have been replaced by `BRIDGE_ENTER`/`BRIDGE_EXIT`. The `BRIDGE_ENTER` and `BRIDGE_EXIT` events are raised when a channel enters/exits any bridge, regardless of whether or not that bridge happens to contain multiple parties.

CLI

- When compiled with `--enable-dev-mode`, the `astobj2` library will now add several CLI commands that allow for inspection of ao2 containers that register themselves with `astobj2`. The CLI commands are:
 - `astobj2 container dump`
 - `astobj2 container stats`
 - `astobj2 container check`
- Added specific CLI commands for bridge inspection. This includes:
 - `bridge show all` - list all bridges in the system
 - `bridge show {id}` - provide specific information about a bridge
- Added CLI command `bridge destroy`. This will destroy the specified bridge, ejecting the channels currently in the bridge. If the channels cannot continue in the dialplan or application that put them in the bridge, they will be hung up.
- Added command `bridge kick`. This will eject a single channel from a bridge.
- Added commands to inspect and manipulate the registered bridge technologies. This includes:
 - `bridge technology show` - list the registered bridge technologies
 - `bridge technology {suspend|unsuspend} {tech}` - control whether or not a registered bridge technology can be used during smart bridge operations. If a technology is suspended, it will not be used when a bridge technology is picked for channels; when unsuspended, it can be used again.
- The command `config show help` will show configuration documentation for modules with XML configuration documentation. This takes in three optional parameters - `module`, `type`, and `option` - which, when provided, provides greater detail.
 - When `module`, `type`, and `option` are omitted, a listing of all modules with registered documentation is displayed.
 - When `module` is specified, a listing of all configuration types for that module is displayed, along with their synopsis.
 - When `module` and `type` are specified, a listing of all configuration options for that type are displayed along with their synopsis.
 - When `module`, `type`, and `option` are specified, detailed information for that configuration option is displayed.
- Added `core show sounds` and `core show sound` CLI commands. These display a listing of all installed media sounds available on the system and detailed information about a sound, respectively.
- `xmldoc dump` has been added. This CLI command will dump the XML documentation DOM as a string to the specified file. The Asterisk core will populate certain XML elements pulled from the source files with additional run-time information; this command lets a user produce the XML documentation with all information.

Features

- Parking has been pulled from core and placed into a separate module called [res_parking](#). Configuration for parking should now be performed in [res_parking.conf](#). Configuration for parking in [features.conf](#) is now unsupported.
- Core attended transfers now have several new options. While performing an attended transfer, the transferer now has the following options:
 - *1 - cancel the attended transfer (configurable via `atxferabort`)
 - *2 - complete the attended transfer, dropping out of the call (configurable via `atxfercomplete`)
 - *3 - complete the attended transfer, but stay in the call. This will turn the call into a multi-party bridge (configurable via `atxfert hreeway`)
 - *4 - swap to the other party. Once an attended transfer has begun, this options may be used multiple times (configurable via `atxferswap`)
- For DTMF blind and attended transfers, the channel variable `TRANSFER_CONTEXT` must be on the channel initiating the transfer to have any effect.
- The `BRIDGE_FEATURES` channel variable would previously only set features for the calling party and would set this feature regardless of whether the feature was in caps or in lowercase. Use of a caps feature for a letter will now apply the feature to the calling party while use of a lowercase letter will apply that feature to the called party.
- Add support for `automixmon` to the `BRIDGE_FEATURES` channel variable.
- The channel variable `DYNAMIC_PEERNAME` is redundant with `BRIDGEPEER` and is removed. The more useful `DYNAMIC_WHO_ACTIVATE`

D gives the channel name that activated the dynamic feature.

- The channel variables `DYNAMIC_FEATURENAME` and `DYNAMIC_WHO_ACTIVATED` are set only on the channel executing the dynamic feature. Executing a dynamic feature on the bridge peer in a multi-party bridge will execute it on all peers of the activating channel.
- You can now have the settings for a channel updated using the `FEATURE()` and `FEATUREMAP()` functions inherited to child channels by setting `FEATURE(inherit)=yes`.
- `automixmon` now supports additional channel variables from `automon` including: `TOUCH_MIXMONITOR_PREFIX`, `TOUCH_MIXMONITOR_MESSAGE_START`, and `TOUCH_MIXMONITOR_MESSAGE_STOP`.
- A new general `features.conf` option `recordingfailsound` has been added which allows setting a failure sound for a user tries to invoke a recording feature such as `automon` or `automixmon` and it fails.
- It is no longer necessary (or possible) to define the `ATXFER_NULL_TECH` in `features.c` for `atxferdropcall=no` to work properly. This option now just works.

Logging

- Added log rotation strategy `none`. If set, no log rotation strategy will be used. Given that this can cause the Asterisk log files to grow quickly, this option should only be used if an external mechanism for log management is preferred.

Realtime

- Dynamic realtime tables for SIP Users can now include a `path` field. This will store the path information for that peer when it registers. Realtime tables can also use the `supportpath` field to enable Path header support.
- LDAP realtime configurations for SIP Users now have the `AstAccountPathSupport` `objectIdentifier`. This maps to the `supportpath` option in `sip.conf`.

Sorcery

- Sorcery is a new data abstraction and object persistence API in Asterisk. It provides modules a useful abstraction on top of the many storage mechanisms in Asterisk, including the Asterisk Database, static configuration files, static Realtime, and dynamic Realtime. It also provides a caching service. Users can configure a hierarchy of data storage layers for specific modules in `sorcery.conf`.
- All future modules which utilize Sorcery for object persistence must have a column named `id` within their schema when using the Sorcery realtime module. This column must be able to contain a string of up to 128 characters in length.

Security Events Framework

- Security Event timestamps now use ISO 8601 formatted date/time instead of the `seconds-microseconds` format that it was using previously.

Stasis Message Bus

- The Stasis message bus is a publish/subscribe message bus internal to Asterisk. Many services in Asterisk are built on the Stasis message bus, including AMI, ARI, CDRs, and CEL. Parameters controlling the operation of Stasis can be configured in `stasis.conf`. Note that these parameters operate at a very low level in Asterisk, and generally will not require changes.

Channel Drivers

- When a channel driver is configured to enable jitterbuffers, they are now applied unconditionally when a channel joins a bridge. If a jitterbuffer is already set for that channel when it enters, such as by the `JITTERBUFFER` function, then the existing jitterbuffer will be used and the one set by the channel driver will not be applied.

chan_agent

- `chan_agent` has been removed and replaced with [AgentLogin](#) and [AgentRequest](#) dialplan applications provided by the `app_agent_pool` module. Agents are connected with callers using the new [AgentRequest](#) dialplan application. The `Agents:<agent-id>` device state is available to monitor the status of an agent. See [agents.conf.sample](#) for valid configuration options.
- The `updatecdr` option has been removed. Altering the names of channels on a CDR is not supported - the name of the channel is the name of the channel, and pretending otherwise helps no one. The `AGENTUPDATECDR` channel variable has also been removed, for the same reason.
- The `endcall` and `enddtmf` configuration options are removed. Use the dialplan function `CHANNEL(dtmf-features)` to set DTMF features on the agent channel before calling [AgentLogin](#).

chan_bridge

- `chan_bridge` has been removed. Its functionality has been incorporated directly into the [ConfBridge](#) application itself.

chan_dahdi

- Added the CLI command `pri destroy span`. This will destroy the D-channel of the specified span and its B-channels. Note that this command should only be used if you understand the risks it entails.
- The CLI command `dahdi destroy channel` is now `dahdi destroy channels`. A range of channels can be specified to be destroyed. Note that this command should only be used if you understand the risks it entails.
- Added the CLI command `dahdi create channels`. A range of channels can be specified to be created, or the keyword `new` can be used to add channels not yet created.
- The script specified by the `chan_dahdi.conf` `mwimonitornotify` option now gets the exact configured mailbox name. For `app_voicemail` mailboxes this is `mailbox@context`.
- Added `mw_i_vm_boxes` that also must be configured for ISDN MWI to be enabled.

chan_iax2

- IPv6 support has been added. `chan_iax2` is now able to bind to and communicate using IPv6 addresses.

chan_local

- The `/b` option has been removed.
- `chan_local` moved into the system core and is no longer a loadable module.

chan_mobile

- Added general support for busy detection.
- Added ECAM command support for Sony Ericsson phones.

chan_pjsip

- A new SIP channel driver for Asterisk, `chan_pjsip` is built on the PJSIP SIP stack. A collection of resource modules provides the bulk of the SIP functionality. For more information on configuring the new SIP channel driver, see the [Configuring res_pjsip](#).

chan_sip

- Added support for [RFC 3327 "Path"](#) headers. This can be enabled in `sip.conf` using the `supportpath` setting, either on a global basis or on a peer basis. This setting enables Asterisk to route outgoing out-of-dialog requests via a set of proxies by using a pre-loaded route-set defined by the Path headers in the `REGISTER` request. See [Realtime](#) updates for more configuration information.
- The `SIP_CODEC` family of variables may now specify more than one codec. Each codec must be separated by a comma. The first codec specified is the preferred codec for the offer. This allows a dialplan writer to specify both audio and video codecs, e.g.,

```
same => n,Set(SIP_CODEC=ulaw,h264)
```


- The `callevnts` parameter has been removed. [Hold](#) AMI events are now raised in the core, and can be filtered out using the `eventfilter` parameter in `manager.conf`.
- Added `ignore_requested_pref`. When enabled, this will use the preferred codecs configured for a peer instead of the requested codec.
- The option `register_retry_403` has been added to `chan_sip` to work around servers that are known to erroneously send 403 in response to valid REGISTER requests and allows Asterisk to continue attempting to connect.

chan_skinny

- Added the `immedialkey` parameter. If set, when the user presses the configured key the already entered number will be immediately dialed. This is useful when the dialplan allows for variable length pattern matching. Valid options are `*` and `#`.
- Added the `callfdtimeout` parameter. This configures the amount of time (in milliseconds) before a call forward is considered to not be answered.
- The `serviceurl` parameter allows Service URLs to be attached to line buttons.

Functions

AGENT

- The password option has been disabled, as the [AgentLogin](#) application no longer provides authentication.

AUDIOHOOK_INHERIT

- Due to changes in the Asterisk core, this function is no longer needed to preserve a [MixMonitor](#) on a channel during transfer operations and dialplan execution. It is effectively obsolete.

CDR (function)

- The `amaflags` and `accountcode` attributes for the [CDR](#) function are deprecated. Use the [CHANNEL](#) function instead to access these attributes.
- The `l` option has been removed. When reading a CDR attribute, the most recent record is always used. When writing a CDR attribute, all non-finalized CDRs are updated.
- The `r` option has been removed, for the same reason as the `l` option.
- The `s` option has been removed, as `LOCKED` semantics no longer exist in the CDR engine.

CDR_PROP

- A new function [CDR_PROP](#) has been added. This function lets you set properties on a channel's active CDRs. This function is write-only. Properties accept boolean values to set/clear them on the channel's CDRs. Valid properties include:
 - `party_a` - make this channel the preferred Party A in any CDR between two channels. If two channels have this property set, the creation time of the channel is used to determine who is Party A. Note that dialed channels are ever Party A in a CDR.
 - `disable` - disable CDRs on this channel. This is analogous to the [NoCDR](#) application when set to `True`, and analogous to the `e` option in [ResetCDR](#) when set to `False`.

CHANNEL

- Added the argument `dtmf_features`. This sets the DTMF features that will be enabled on a channel when it enters a bridge. Allowed values are `T`, `K`, `H`, `W`, and `X`, and are analogous to the parameters passed to the [Dial](#) application.
- Added the argument `after_bridge_goto`. This can be set to a parseable Goto string, i.e., `[[context],extension],priority`. When set, if a channel leaves a bridge but is not hung up it will resume dialplan execution at that location.

JITTERBUFFER

- JITTERBUFFER now accepts an argument of `disabled` which can be used to remove jitterbuffers previously set on a channel with JITTERBUFFER. The value of this setting is ignored when `disabled` is used for the argument.

PJSIP_DIAL_CONTACTS

- A new function provided by `chan_pjsip`, this function can be used in conjunction with the [Dial](#) application to construct a dial string that will dial all contacts on an Address of Record associated with a `chan_pjsip` endpoint.

PJSIP_MEDIA_OFFER

- Provided by `chan_pjsip`, this function sets the codecs to be offered on the outbound channel prior to dialing.

REDIRECTING

- Redirecting reasons can now be set to arbitrary strings. This means that the REDIRECTING dialplan function can be used to set the redirecting reason to any string. It also allows for custom strings to be read as the redirecting reason from SIP Diversion headers.

SPEECH_ENGINE

- The SPEECH_ENGINE function now supports read operations. When read from, it will return the current value of the requested attribute.

VMCOUNT

- Mailboxes defined by `app_voicemail` **MUST** be referenced by the rest of the system as `mailbox@context`. The rest of the system cannot add `@default` to mailbox identifiers for `app_voicemail` that do not specify a context any longer. It is a mailbox identifier format that should only be interpreted by `app_voicemail`.

Resources

res_agi (Asterisk Gateway Interface)

- The manager event AGIExec has been split into [AGIExecStart](#) and [AGIExecEnd](#).
- The manager event AsyncAGI has been split into [AsyncAGIStart](#), [AsyncAGIExec](#), and [AsyncAGIEnd](#).
- The [CONTROL STREAM FILE](#) command now accepts an `offsetms` parameter. This will start the playback of the audio at the position specified. It will also return the final position of the file in `endpos`.
- The [CONTROL STREAM FILE](#) command will now populate the `CPLAYBACKSTATUS` channel variable if the user stopped the file playback or if a remote entity stopped the playback. If neither stopped the playback, it will indicate the overall success/failure of the playback. If stopped early, the final offset of the file will be set in the `CPLAYBACKOFFSET` channel variable.
- The [SAY ALPHA](#) command now accepts an additional parameter to control whether it specifies the case of uppercase, lowercase, or all letters to provide functionality similar to [SayAlphaCase](#).

res_ari (Asterisk REST Interface) (and others)

- The [Asterisk REST Interface \(ARI\)](#) provides a mechanism to expose and control telephony primitives in Asterisk by remote client. This includes channels, bridges, endpoints, media, and other fundamental concepts. Users of ARI can develop their own communications applications, controlling multiple channels using an HTTP REST interface and receiving JSON events about the objects via a WebSocket connection. ARI can be configured in Asterisk via [ari.conf](#).

res_parking

- Parking has been extracted from the Asterisk core as a loadable module, `res_parking`. Configuration for parking is now provided by [res_parking.conf](#). Configuration through `features.conf` is no longer supported.



`res_parking` uses the configuration framework. If an invalid configuration is supplied, `res_parking` will fail to load or fail to reload. Previously, invalid configurations would generally be accepted, with certain errors resulting in individually disabled parking lots.

- Parked calls are now placed in bridges. While this is largely an architectural change, it does have implications on how channels in a parking lot are viewed. For example, commands that display channels in bridges will now also display the channels in a parking lot.
- The order of arguments for the new parking applications have been modified. Timeout and return `context/exten/priority` are now implemented as options, while the name of the parking lot is now the first parameter. See the application documentation for [Park](#), [Parked Call](#), and [ParkAndAnnounce](#) for more in-depth information as well as syntax.
- Extensions are by default no longer automatically created in the dialplan to park calls or pickup parked calls. Generation of dialplan extensions can be enabled using the `parkext` configuration option.
- ADSI functionality for parking is no longer supported. The `adsipark` configuration option has been removed as a result.
- The `PARKINGSLLOT` channel variable has been deprecated in favor of `PARKING_SPACE` to match the naming scheme of the new system.
- `PARKING_SPACE` and `PARKEDLOT` channel variables will now be set for a parked channel even when the configuration option `comebacktoorigin` is enabled.
- A new CLI command `parking show` has been added. This allows a user to inspect the parking lots that are currently in use. `parking show <parkinglot>` will also show the parked calls in a specific parking lot.
- The CLI command `parkedcalls` is now deprecated in favor of `parking show <parkinglot>`.
- The AML command [ParkedCalls](#) will now accept a *ParkingLot* argument which can be used to get a list of parked calls for a specific parking lot.
- The AML command [Park](#) field *Channel2* has been deprecated and replaced with *TimeoutChannel*. If both *Channel2* and *TimeoutChannel* are specified, *TimeoutChannel* will be used. The field *TimeoutChannel* is no longer a required argument.
- The [ParkAndAnnounce](#) application is now provided through `res_parking` instead of through the separate `app_parkandannounce` module.
- [ParkAndAnnounce](#) will no longer go to the next position in dialplan on timeout by default. Instead, it will follow the timeout rules of the parking lot. The old behavior can be reproduced by using the `c` option.
- Dynamic parking lots will now fail to be created under the following conditions:
 - If the parking lot specified by `PARKINGDYNAMIC` does not exist.
 - If they require exclusive park and parked call extensions which overlap with existing parking lots.
- Dynamic parking lots will be cleared on reload for dynamic parking lots that currently contain no calls. Dynamic parking lots containing parked calls will persist through the reloads without alteration.
- If `parkext_exclusive` is set for a parking lot and that extension is already in use when that parking lot tries to register it, this is now considered a parking system configuration error. Configurations which do this will be rejected.
- Added channel variable `PARKER_FLAT`. This contains the name of the extension that would be used if `comebacktoorigin` is enabled. This can be useful when `comebacktoorigin` is disabled, but the dialplan or an external control mechanism wants to use the extension in the park-dial context that was generated to re-dial the parker on timeout.

res_pjsip (and many others)

- A large number of resource modules make up the SIP stack based on PJSIP. The `chan_pjsip` channel driver uses these resource modules to provide various SIP functionality in Asterisk. The majority of configuration for these modules is performed in [pjsip.conf](#). Other modules may use their own configuration files. See [Asterisk 12 Module Configuration](#) for more information.

res_rtp_asterisk

- ICE/STUN/TURN support in `res_rtp_asterisk` has been made optional. To enable them, the Asterisk-specific version of PJSIP should now be installed. Tarballs are available from <https://github.com/asterisk/pjproject/tags/>.

res_statsd/res_chan_stats

- A new resource module, `res_statsd`, has been added, which acts as a [statsd](#) client. This module allows Asterisk to publish statistics to a statsd server. In conjunction with `res_chan_stats`, it will publish statistics about Asterisk channels to the statsd server. It can be configured via [res_statsd.conf](#).

res_xmpp

- Device state for XMPP buddies is now available using the following format:

```
XMPP/<client name>/<buddy address>
```

If any resource is available the device state is considered to be not in use. If no resources exist or all are unavailable the device state is considered to be unavailable.

Scripts

Realtime/Database Scripts

- Asterisk previously included example db schemas in the `contrib/realtime/` directory of the source tree. This has been replaced by a set of database migrations using the [Alembic framework](#). This allows you to use Alembic to initialize the database for you. It will also serve as a database migration tool when upgrading Asterisk in the future. See [contrib/ast-db-manage/README.md](#) for more details.

sip_to_res_pjsip.py

- A new script has been added in the `contrib/scripts/sip_to_res_pjsip` folder. This python script will convert an existing *sip.conf* file to a *pjsip.conf* file, for use with the `chan_pjsip` channel driver. This script is meant to be an aid in converting an existing `chan_sip` configuration to a `chan_pjsip` configuration, but it is expected that configuration beyond what the script provides will be needed.

Upgrading to Asterisk 12



There are many significant architectural changes in Asterisk 12. It is recommended that you not only read through this document for important changes that affect an upgrade, but that you also read through the [CHANGES](#) document and the information about what is [New in 12](#) to better understand the new options available to you.

Upgrade Overview

Of particular note, the following systems in Asterisk underwent significant changes. Documentation for the changes and a specification for their behavior in Asterisk 12 are available in the [Asterisk 12 Documentation](#) section on this wiki.

- **AMI:** Many events were changed, and the semantics of channels and bridges were defined. In particular, how channels and bridges behave under transfer scenarios and situations involving multiple parties has changed significantly. See the [AMI v2 Specification](#) for more information.
- **CDRs:** CDR logic was extracted from the many locations it existed in across Asterisk and implemented as a consumer of Stasis message bus events. As a result, consistency of records has improved significantly and the behavior of CDRs in transfer scenarios has been defined in the CDR specification. However, significant behavioral changes in CDRs resulted from the transition. The most significant change is the addition of CDR entries when a channel who is the Party A in a CDR leaves a bridge. See the [Asterisk 12 CDR Specification](#) for more information.
- **CEL:** Much like CDRs, CEL was removed from the many locations it existed in across Asterisk and implemented as a consumer of Stasis message bus events. It now closely follows the Bridging API model of channels and bridges, and has a much closer consistency of conveyed events as AMI. For the changes in events, see the [Asterisk 12 CEL Specification](#).

Upgrade Changes

Build System:

- Removed the CHANNEL_TRACE development mode build option. Certain aspects of the CHANNEL_TRACE build option were incompatible with the new bridging architecture.
- Asterisk now depends on `libjansson`, `libuuid` and optionally (but recommended) `libxslt` and `uriparser`.
- The new SIP stack and channel driver uses a particular version of PJSIP. Please see [Installing pjsip](#) for more information on installing and configuring PJSIP for use with Asterisk 12.

AgentLogin, AgentRequest, and chan_agent:

- Along with [AgentRequest](#), this application has been modified to be a replacement for `chan_agent`. The `chan_agent` module and the Agent channel driver have been removed from Asterisk, as the concept of a channel driver proxying in front of another channel driver was incompatible with the new architecture (and has had numerous problems through past versions of Asterisk). The act of a channel calling the [AgentLogin](#) application places the channel into a pool of agents that can be requested by the AgentRequest application. Note that this application, as well as all other agent related functionality, is now provided by the `app_agent_pool` module.
- This application no longer performs agent authentication. If authentication is desired, the dialplan needs to perform this function using the [Authenticate](#) or [VMAuthenticate](#) application or through an AGI script before running AgentLogin.
- The `agents.conf` schema has changed. Rather than specifying agents on a single line in comma delineated fashion, each agent is defined in a separate context. This allows agents to use the power of context templates in their definition.
- A number of parameters from `agents.conf` have been removed. This includes `maxloginretries`, `autologoffunavail`, `updatecdr`, `goodbye`, `group`, `recordformat`, `urlprefix`, and `savecallsin`. These options were obsoleted by the move from a channel driver model to the bridging/application model provided by `app_agent_pool`.
- The `AGENTUPDATECDR` channel variable has also been removed, for the same reason as the `updatecdr` option.
- The `endcall` and `enddtmf` configuration options are removed. Use the dialplan function `CHANNEL(dtmf-features)` to set DTMF features on the agent channel before calling AgentLogin.

AgentMonitorOutgoing

- This application has been removed. It was a holdover from when AgentCallbackLogin was removed.

Answer

- It is no longer possible to bypass updating the CDR when answering a channel. CDRs are based on the channel state and will be updated when the channel is Answered.

ControlPlayback

- The channel variable `CPLAYBACKSTATUS` may now return the value `REMOTESTOPPED` when playback is stopped by an external entity.

DISA

- This application now has a dependency on the `app_cdr` module. It uses this module to hide the CDR created prior to execution of the DISA application.

DumpChan:

- The output of DumpChan no longer includes the *DirectBridge* or *IndirectBridge* fields. Instead, if a channel is in a bridge, it includes a *BridgeID* field containing the unique ID of the bridge that the channel happens to be in.

ForkCDR:

- Nearly every parameter in ForkCDR has been updated and changed to reflect the changes in CDRs. Please see the documentation for the [ForkCDR](#) application, as well as the [CDR specification](#).

NoCDR:

- The NoCDR application has been deprecated. Please use the `CDR_PROP` function to disable CDRs on a channel.

ParkAndAnnounce:

- The `app_parkandannounce` module has been removed. The application ParkAndAnnounce is now provided by the `res_parking` module. See the [Parking](#) changes for more information.

ResetCDR:

- The `w` and `a` options have been removed. Dispatching CDRs to registered backends occurs on an as-needed basis in order to preserve linkedid propagation and other needed behavior.
- The `e` option is deprecated. Please use the `CDR_PROP` function to enable CDRs on a channel that they were previously disabled on.
- The [ResetCDR](#) application is no longer a part of core Asterisk, and instead is now delivered as part of `app_cdr`.

Queues:

- Queue strategy `rrmemory` now has a predictable order similar to strategy `rrordered`. Members will be called in the order that they are added to the queue.
- Removed the `queues.conf` `check_state_unknown` option. It is no longer necessary.
- It is now possible to play the Queue prompts to the first user waiting in a call queue. Note that this may impact the ability for agents to talk with users, as a prompt may still be playing when an agent connects to the user. This ability is disabled by default but can be enabled on an individual queue using the `announce-to-first-user` option.
- The configuration options `eventwhencalled` and `eventmemberstatus` have been removed. As a result, the AMI events [QueueMemberStatus](#), [AgentCalled](#), [AgentConnect](#), [AgentComplete](#), [AgentDump](#), and [AgentRingNoAnswer](#) will always be sent. The *Variable* fields will also no longer exist on the *Agent** events. These events can be filtered out from a connected AMI client using the `eventfilter` setting

in `manager.conf`.

- The queue log now differentiates between blind and attended transfers. A blind transfer will result in a `BLINDTRANSFER` message with the destination context and extension. An attended transfer will result in an `ATTENDEDTRANSFER` message. This message will indicate the method by which the attended transfer was completed: `BRIDGE` for a bridge merge, `APP` for running an application on a bridge or channel, or `LINK` for linking two bridges together with local channels. The queue log will also now detect externally initiated blind and attended transfers and record the transfer status accordingly.
- When performing queue pause/unpause on an interface without specifying an individual queue, the `PAUSEALL/UNPAUSEALL` event will only be logged if at least one member of any queue exists for that interface.

SetAMAFlags

- This application is deprecated in favor of `CHANNEL(amaflags)`.

VoiceMail:

- The `voicemail.conf` configuration file now has an `alias` configuration parameter for use with the `Directory` application. The voicemail realtime database table schema has also been updated with an 'alias' column. Systems using voicemail with realtime should update their schemas accordingly.
- Mailboxes defined by `app_voicemail` **MUST** be referenced by the rest of the system as `mailbox@context`. The rest of the system cannot add `@default` to mailbox identifiers for `app_voicemail` that do not specify a context any longer. It is a mailbox identifier format that should only be interpreted by `app_voicemail`.

Channel Drivers:

- When a channel driver is configured to enable jitterbuffers, they are now applied unconditionally when a channel joins a bridge. If a jitterbuffer is already set for that channel when it enters, such as by the `JITTERBUFFER` function, then the existing jitterbuffer will be used and the one set by the channel driver will not be applied.

chan_bridge

- `chan_bridge` is removed and its functionality is incorporated into `ConfBridge` itself.

chan_dahdi:

- Analog port dialing and deferred DTMF dialing for PRI now distinguishes between `w` and `W`. The `w` pauses dialing for half a second. The `W` pauses dialing for one second.
- The default for `inband_on_proceeding` has changed to `no`.
- The CLI command `dahdi destroy channel` is now `dahdi destroy channels`. A range of channels can be specified to be destroyed. Note that this command should only be used if you understand the risks it entails.
- The script specified by the `chan_dahdi.conf` `mwimonitornotify` option now gets the exact configured mailbox name. For `app_voicemail` mailboxes this is `mailbox@context`.
- Added `mw_i_vm_boxes` that also must be configured for ISDN MWI to be enabled.

chan_local:

- The `/b` option has been removed.
- `chan_local` moved into the system core and is no longer a loadable module.

chan_sip:

- The `calleevents` parameter has been removed. `Hold` AML events are now raised in the core, and can be filtered out using the `eventfilter` parameter in `manager.conf`.
- Dynamic realtime tables for SIP Users can now include a `path` field. This will store the path information for that peer when it registers. Realtime tables can also use the `supportpath` field to enable Path header support.

- LDAP realtime configurations for SIP Users now have the `AstAccountPathSupport` `objectIdentifier`. This maps to the `supportpath` option in `sip.conf`.

Core:

- Masquerades as an operation inside Asterisk have been effectively hidden by the migration to the Bridging API. As such, many 'quirks' of Asterisk no longer occur. This includes renaming of channels, "<ZOMBIE>" channels, dropping of frame/audio hooks, and other internal implementation details that users had to deal with. This fundamental change has large implications throughout the changes documented for this version. For more information about the new core architecture of Asterisk, please see the Asterisk wiki.
- The following channel variables have changed behavior which is described in the CHANGES file: `TRANSFER_CONTEXT`, `BRIDGEPEER`, `BRIDGEPVTCALLID`, `ATTENDED_TRANSFER_COMPLETE_SOUND`, `DYNAMIC_FEATURENAME`, and `DYNAMIC_PEERNAME`.
- All bridging in Asterisk is now performed using the Bridging API, which is the same bridging core that powers the `ConfBridge` application. As a result, significant changes have been made in all areas of Asterisk that are affected by bridging. Those changes are noted in the appropriate areas.

AMI (Asterisk Manager Interface):

- The Version has been increased to 1.4. For a full listing of the semantics changes in AMI, see the [AMI v2 Specification](#).
- The details of what happens to a channel when a masquerade happens (transfers, parking, etc) have changed. Channels no longer swap Uniqueid's as a result of the masquerade. In general, AMI clients will never actually "see" a masquerade, as the operation has been effectively hidden from external systems.
- **Major** changes were made to both the syntax as well as the semantics of the AMI protocol. In particular, AMI events have been substantially modified and improved in this version of Asterisk. The major event changes are listed below:
 - `NewPeerAccount` has been removed. `NewAccountCode` is raised instead.
 - `Reload` events have been consolidated and standardized.
 - `ModuleLoadReport` has been removed.
 - `FaxSent` is now `SendFAX`; `FaxReceived` is now `ReceiveFAX`. This standardizes `app_fax` and `res_fax` events.
 - `MusicOnHold` has been replaced with `MusicOnHoldStart` and `MusicOnHoldStop`.
 - `JabberEvent` has been removed.
 - `Hold` is now in the core and will now raise `Hold` and `Unhold` events.
 - `Join` is now `QueueCallerJoin`.
 - `Leave` is now `QueueCallerLeave`.
 - `Agentlogin/Agentlogoff` is now `AgentLogin/AgentLogoff`, respectively.
 - `ChannelUpdate` has been removed.
 - Local channel optimization is now conveyed via `LocalOptimizationBegin` and `LocalOptimizationEnd`.
 - `BridgeAction` and `BridgeExec` have been removed.
 - `BlindTransfer` and `AttendedTransfer` events were added.
 - `Dial` is now `DialBegin` and `DialEnd`.
 - `DTMF` is now `DTMFBegin` and `DTMFEnd`.
 - `Bridge` has been replaced with `BridgeCreate`, `BridgeEnter`, `BridgeLeave`, and `BridgeDestroy`.
 - The `Masquerade` event is no longer raised. The `Rename` event still exists, but only silly channel drivers (`chan_misdn`) will ever think of raising it, and you shouldn't be using `chan_misdn` anyway.
 - `MusicOnHold` has been replaced with `MusicOnHoldStart` and `MusicOnHoldStop`.
 - `AGIExec` is now `AGIExecStart` and `AGIExecEnd`.
 - `AsyncAGI` is now `AsyncAGIStart`, `AsyncAGIExec`, and `AsyncAGIEnd`.
- The `MCID` AMI event now publishes a channel snapshot when available and its non-channel-snapshot parameters now use either the `M CallerID` or `MConnectedID` prefixes with `Subaddr*`, `Name*`, and `Num*` suffixes instead of `CallerID` and `ConnectedID` to avoid confusion with similarly named parameters in the channel snapshot.
- The `Channel` key used in the `AlarmClear`, `Alarm`, and `DNDState` has been renamed `DAHDIChannel` since it does not convey an Asterisk channel name.
- All AMI events now contain a `SystemName` field, if available.
- Local channel information in events is now prefixed with `LocalOne` and `LocalTwo`. This replaces the suffix of '1' and '2' for the two halves of the Local channel. This affects the `LocalBridge`, `LocalOptimizationBegin`, and `LocalOptimizationEnd` events.
- The `RTCPSent/RTCPReceived` events have been significantly modified from previous versions. They now report all SR/RR packets sent/received, and have been restructured to better reflect the data sent in a SR/RR. In particular, the event structure now supports multiple report blocks.
- The deprecated use of `|` (pipe) as a separator in the `channelvars` setting in `manager.conf` has been removed.
- The SIP `SIPqualifypeer` action now sends a response indicating it will qualify a peer once a peer has been found to qualify. Once the

qualify has been completed it will now issue a [SIPqualifypeerdone](#) event.

- The AMI event [Newexten](#) field *Extension* is deprecated, and may be removed in a future release. Please use the common *Exten* field instead.
- The AMI events [ParkedCall](#), [ParkedCallTimeOut](#), [ParkedCallGiveUp](#), and [UnParkedCall](#) have changed significantly in the new `res_parking` module.
 - The *Channel* and *From* headers are gone. For the channel that was parked or is coming out of parking, a *Parkee* channel snapshot is issued and it has a number of fields associated with it. The old *Channel* header relayed the same data as the new *ParkeeChannel* header.
 - The *From* field was ambiguous and changed meaning depending on the event. For most of these, it was the name of the channel that parked the call (the *Parker*). There is no longer a header that provides this channel name, however the *ParkerDialString* will contain a dialstring to redial the device that parked the call.
 - On [UnParkedCall](#) events, the *From* header would instead represent the channel responsible for retrieving the *parkee*. It receives a channel snapshot labeled *Retriever*. The *from* field is replaced with *RetrieverChannel*.
 - Lastly, the *Exten* field has been replaced with *ParkingSpace*.
- The AMI event [Parkinglot](#) (response to [Parkinglots](#) command) in a similar fashion has changed the field names *StartExten* and *StopExten* to *StartSpace* and *StopSpace* respectively.
- The AMI [Status](#) response event to the AMI [Status](#) action replaces the *BridgedChannel* and *BridgedUniqueid* headers with the *BridgeID* header to indicate what bridge the channel is currently in.

CDR (Call Detail Records)

- Significant changes have been made to the behavior of CDRs. The CDR engine was effectively rewritten and built on the Stasis message bus. For a full definition of CDR behavior in Asterisk 12, please read the [Asterisk 12 CDR Specification](#).
- CDRs will now be created between all participants in a bridge. For each pair of channels in a bridge, a CDR is created to represent the path of communication between those two endpoints. This lets an end user choose who to bill for what during bridge operations with multiple parties.
- The duration, billsec, start, answer, and end times now reflect the times associated with the current CDR for the channel, as opposed to a cumulative measurement of all CDRs for that channel.
- CDR backends can no longer be unloaded while billing data is in flight. This helps to prevent loss of billing data during restarts and shutdowns.

CEL:

- The *Uniqueid* field for a channel is now a stable identifier, and will not change due to transfers, parking, etc.
- CEL has undergone significant rework in Asterisk 12, and is now built on the Stasis message bus. Please see the [Asterisk 12 CEL Specification](#) for more detailed information. A summary of the affected events is below:
 - `BRIDGE_START`, `BRIDGE_END`, `BRIDGE_UPDATE`, `3WAY_START`, `3WAY_END`, `CONF_ENTER`, `CONF_EXIT`, `CONF_START`, and `CONF_END` events have all been removed. These events have been replaced by `BRIDGE_ENTER`/`BRIDGE_EXIT`.
 - `BLINDTRANSFER`/`ATTENDEDTRANSFER` events now report the peer as `NULL` and additional information in the extra string field.

Dialplan:

- All channel and global variable names are evaluated in a case-sensitive manner. In previous versions of Asterisk, variables created and evaluated in the dialplan were evaluated case-insensitively, but built-in variables and variable evaluation done internally within Asterisk was done case-sensitively.
- Asterisk has always had code to ignore dash '-' characters that are not part of a character set in the dialplan extensions. The code now consistently ignores these characters when matching dialplan extensions.
- `BRIDGE_FEATURES` channel variable is now case sensitive for feature letter codes. Uppercase variants apply them to the calling party while lowercase variants apply them to the called party.

Dialplan Functions:

- Certain dialplan functions have been marked as 'dangerous', and may only be executed from the dialplan. Execution from external sources (AMI's GetVar and SetVar actions; etc.) may be inhibited by setting `live_dangerously` in the `[options]` section of `asterisk.conf` to no. `SHELL()`, channel locking, and direct file read/write functions are marked as dangerous. `DB_DELETE()` and `REALTIME_DESTROY()` are marked as dangerous for reads, but can now safely accept writes (which ignore the provided value).

Features:

- The `features.conf` `[applicationmap] <FeatureName> ActivatedBy` option is no longer honored. The feature is always activated by the channel that has `DYNAMIC_FEATURES` defined on it when it enters the bridge. Use `predial` to set different values of `DYNAMIC_FEATURES` on the channels
- Executing a dynamic feature on the bridge peer in a multi-party bridge will execute it on all peers of the activating channel.
- There is no longer an explicit `features reload` CLI command. Features can still be reloaded using `module reload features`.
- It is no longer necessary (or possible) to define the `ATXFER_NULL_TECH` in `features.c` for `atxferdropcall=no` to work properly. This option now just works.

Parking:

- Parking has been extracted from the Asterisk core as a loadable module, `res_parking`.
- Configuration is found in `res_parking.conf`. It is no longer supported in `features.conf`
- The arguments for the `Park`, `ParkedCall`, and `ParkAndAnnounce` applications have been modified significantly. See the application documents for specific details.
- Numerous changes to Parking related applications, AMI and CLI commands and internal inter-workings have been made. Please read the CHANGES file or the [New in 12](#) page for the detailed list.

Security Events Framework:

- Security Event timestamps now use ISO 8601 formatted date/time instead of the "seconds-microseconds" format that it was using previously.

AGENT:

- The password option has been disabled, as the `AgentLogin` application no longer provides authentication.

AUDIOHOOK_INHERIT:

- Due to changes in the Asterisk core, this function is no longer needed to preserve a `MixMonitor` on a channel during transfer operations and dialplan execution. It is effectively obsolete.

CDR: (function)

- The `amaflags` and `accountcode` attributes for the CDR function are deprecated. Use the `CHANNEL` function instead to access these attributes.
- The `l` option has been removed. When reading a CDR attribute, the most recent record is always used. When writing a CDR attribute, all non-finalized CDRs are updated.
- The `r` option has been removed, for the same reason as the `l` option.
- The `s` option has been removed, as `LOCKED` semantics no longer exist in the CDR engine.

VMCOUNT:

- Mailboxes defined by `app_voicemail` **MUST** be referenced by the rest of the system as `mailbox@context`. The rest of the system cannot add `@default` to mailbox identifiers for `app_voicemail` that do not specify a context any longer. It is a mailbox identifier format that should only be interpreted by `app_voicemail`.

res_rtp_asterisk:

- ICE/STUN/TURN support in `res_rtp_asterisk` has been made optional. To enable them, an Asterisk-specific version of PJSIP needs to be installed. Tarballs are available from <https://github.com/asterisk/pjproject/tags/>.

Asterisk 12 Installation and Configuration

Overview

Asterisk 12 is installed and configured in much the same way as previous versions of Asterisk - for more information, please see [Installing Asterisk](#) and [Configuration and Operation](#). However, some of the new capabilities of Asterisk 12, notably the new chan_pjsip channel driver based on [pjproject](#) as well as the [Alembic](#) database migration tool, require new installation and configuration procedures. To learn more, please review the pages contained in this section.

Installing pjproject

- [Overview](#)
- [Building and Installing pjproject from Source](#)
 - [Downloading pjproject](#)
 - [Building and Installing pjproject](#)
 - [Issues and Workarounds](#)
- [Uninstalling a Previous Version of pjproject](#)

Overview

Asterisk 12 contains two SIP stacks: one, the original chan_sip SIP channel driver that has been present in all previous releases of Asterisk, and a new SIP stack that is based on [pjproject](#). For background information on the decision to write a new SIP channel driver for Asterisk 12, please read the [New SIP Channel Driver](#) page.

Because the current release of pjproject cannot build shared object libraries, some changes were required in order to use it with Asterisk 12. As a result, the current versions of pjproject that can be downloaded from www.pjsip.org will **not** work with Asterisk 12.



Unlike Asterisk 11, which uses an embedded pjproject for the ICE, STUN and TURN libraries in its RTP engine for WebSockets support, Asterisk 12 dynamically links to pjproject.



If you have previously installed a version of pjproject, you **must** remove that version of pjproject prior to building and installing the Asterisk 12 compatible version of pjproject. See [Uninstalling pjproject](#) for more information.



The Asterisk development team is currently working with Teluu, the makers of pjproject, to push changes upstream so that one day a special version of pjproject won't be required for Asterisk 12.

The Asterisk 12 compatible version of pjproject is available on [github](#), or - depending on your Linux distribution - available as a package. This wiki page provides detailed instructions on building and installing pjproject for Asterisk 12.

Building and Installing pjproject from Source

Downloading pjproject

1. If you do not have [git](#), install git on your local machine.



Downloading and installing git is beyond the scope of these instructions, but for Debian/Ubuntu systems, it should be as simple as:

```
apt-get install git
```

And for RedHat/CentOS systems:

```
yum install git
```

2. Checkout the Asterisk 12-compatible pjproject from the Asterisk [github repo](#):

```
# git clone https://github.com/asterisk/pjproject pjproject
```

And that's it!

Building and Installing pjproject

1. Change directories to the pjproject source directory:

```
# cd pjproject
```


2. In the pjproject source directory, run the configure script:

```
# ./configure --prefix=/usr --enable-shared --disable-sound --disable-resample --disable-video --disable-opencore-amr
```

pjproject embeds a number of third party libraries which can conflict with versions of those libraries that may already be installed on your system. Thus, building pjproject is highly dependent on your distribution of Linux as well as what third party libraries are already installed on your system. A number of configuration options are available to disable these libraries in pjproject and custom tailor it to your system. The table below outlines common ones that may be needed for a typical installation.

Library	Configure option	Notes
libspeex shared objects	--with-external-speex	Make sure that the library development headers are accessible from pjproject. The CFLAGS and LDFLAGS environment variables may be used to set the include/lib paths.
libsrtp shared objects	--with-external-srtp	Make sure that the library development headers are accessible from pjproject. The CFLAGS and LDFLAGS environment variables may be used to set the include/lib paths.
GSM codec	--with-external-gsm	Make sure that the library development headers are accessible from pjproject. The CFLAGS and LDFLAGS environment variables may be used to set the include/lib paths.
Disable sound	--disable-sound	Let Asterisk perform sound manipulations.
Disable resampling	--disable-resample	Let Asterisk perform resample operations.
Disable video	--disable-video	Disable video support in pjproject's media libraries. This is not used by Asterisk.
Disable AMR	--disable-opencore-amr	Disable AMR codec support. This is not used by Asterisk

These are some of the more common options used to disable third party libraries in pjproject. However, other options may be needed depending on your system - see `configure --help` for a full list of configure options you can pass to pjproject.

 You **must** specify `--enable-shared` in order to build the shared objects for Asterisk. `--prefix` should be set to the base path of the `lib` directory where the shared objects will be installed.

 If running on a 64-bit Red Hat distribution (e.g. Fedora, CentOS), then you will also need to add `--libdir=/usr/lib64`

3. Build pjproject:

```
# make dep
# make
```

4. Install pjproject

```
# make install
```

5. Update shared library links.

```
# ldconfig
```

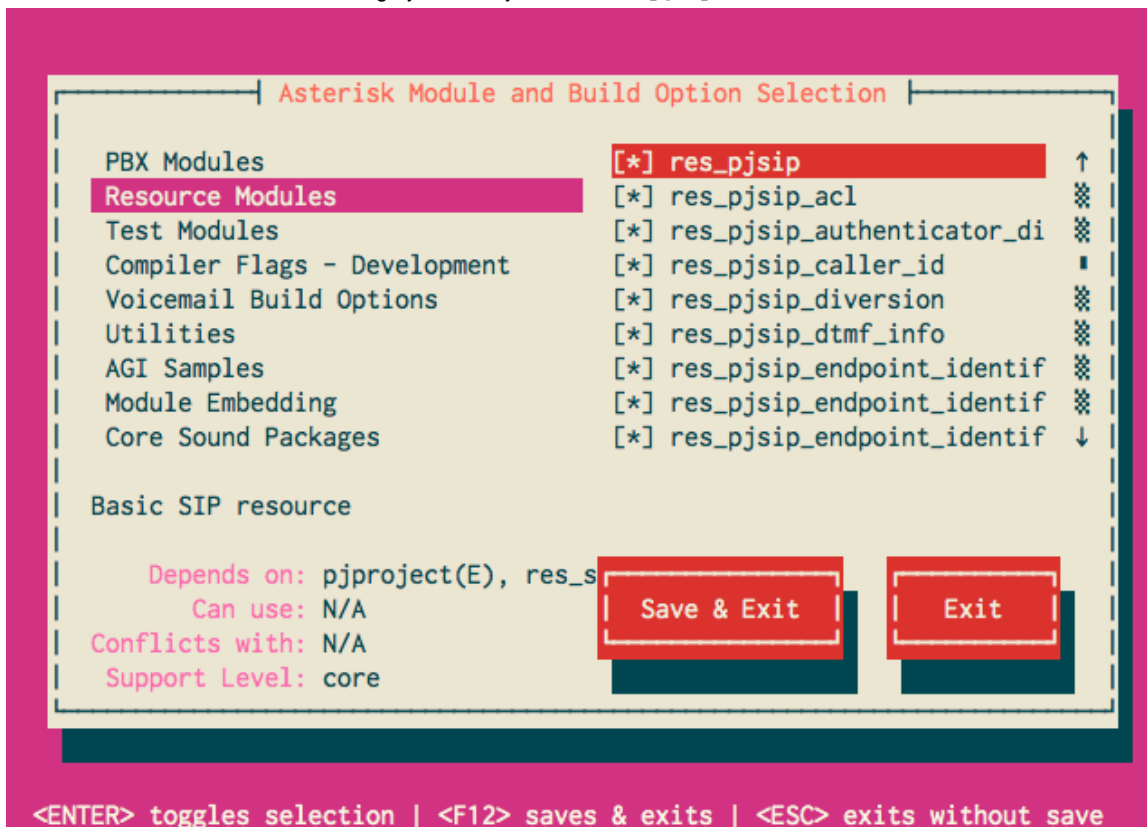
6. Verify that pjproject has been installed in the target location by looking for, and finding the various pjproject modules:

```
# ldconfig -p | grep pj
libpjsua.so (libc6,x86-64) => /usr/lib/libpjsua.so
libpjsip.so (libc6,x86-64) => /usr/lib/libpjsip.so
libpjsip-ua.so (libc6,x86-64) => /usr/lib/libpjsip-ua.so
libpjsip-simple.so (libc6,x86-64) => /usr/lib/libpjsip-simple.so
libpjnath.so (libc6,x86-64) => /usr/lib/libpjnath.so
libpjmedia.so (libc6,x86-64) => /usr/lib/libpjmedia.so
libpjmedia-videodev.so (libc6,x86-64) => /usr/lib/libpjmedia-videodev.so
libpjmedia-codec.so (libc6,x86-64) => /usr/lib/libpjmedia-codec.so
libpjmedia-audiodev.so (libc6,x86-64) => /usr/lib/libpjmedia-audiodev.so
libpjlib-util.so (libc6,x86-64) => /usr/lib/libpjlib-util.so
libpj.so (libc6,x86-64) => /usr/lib/libpj.so
```

7. Finally, verify that Asterisk detects the pjproject libraries. In your Asterisk 12 source directory:

```
# ./configure
# make menuselect
```

8. Browse to the **Resource Modules** category and verify that the `res_pjsip` modules are enabled:



9. You're all done! Now, build and install Asterisk as your normally would.

Issues and Workarounds

Issue

After building and installing pjproject, Asterisk fails to detect any of the libraries - the various `res_pjsip` components cannot be selected in Asterisk's `menuselect`

Workaround

Verify that Asterisk's `config.log` shows the following:

```
configure:23029: checking for PJPROJECT
configure:23036: $PKG_CONFIG --exists --print-errors "libpjproject"
Package libpjproject was not found in the pkg-config search path.
Perhaps you should add the directory containing `libpjproject.pc'
to the PKG_CONFIG_PATH environment variable
No package 'libpjproject' found
```

1. Make sure you have `pkg-config` installed on your system.
2. `pipproject` will install the package config file in `/usr/lib/pkgconfig`. Some distributions, notably Fedora, will instead look for the library in `/usr/lib64`. Update your `PKG_CONFIG_PATH` environment variable with `/usr/lib/pkgconfig` and re-run Asterisk's configure script.

Issue

When building `pipproject`, errors about `opencore_amr` are displayed, e.g.:

```
output/pjmedia-codec-x86_64-unknown-linux-gnu/opencore_amr.o(.rodata+0x60): multiple definition of
`pjmedia_codec_amrnb_framelenbits'
output/pjmedia-codec-x86_64-unknown-linux-gnu/opencore_amr.o(.rodata+0x60): first defined here
output/pjmedia-codec-x86_64-unknown-linux-gnu/opencore_amr.o(.rodata+0x80): multiple definition of
`pjmedia_codec_amrnb_framelen'
output/pjmedia-codec-x86_64-unknown-linux-gnu/opencore_amr.o(.rodata+0x80): first defined here
output/pjmedia-codec-x86_64-unknown-linux-gnu/opencore_amr.o(.rodata+0x20): multiple definition of
`pjmedia_codec_amrwb_framelenbits'
output/pjmedia-codec-x86_64-unknown-linux-gnu/opencore_amr.o(.rodata+0x20): first defined here
output/pjmedia-codec-x86_64-unknown-linux-gnu/opencore_amr.o(.rodata+0x40): multiple definition of
`pjmedia_codec_amrwb_framelen'
output/pjmedia-codec-x86_64-unknown-linux-gnu/opencore_amr.o(.rodata+0x40): first defined here
...
```

Workaround

You already have the AMR codec installed. Run `configure` with the `--disable-opencore-amr` option specified.

Issue

When building `pipproject`, linker errors referring to various video methods are displayed, e.g.:

```
/home/mjordan/projects/pjproject/pjmedia/lib/libpjmedia-videodev.so: undefined reference to `pjmedia_format_init_video'
/home/mjordan/projects/pjproject/pjmedia/lib/libpjmedia.so: undefined reference to `pjmedia_video_format_mgr_instance'
/home/mjordan/projects/pjproject/pjmedia/lib/libpjmedia-videodev.so: undefined reference to
`pjmedia_format_get_video_format_detail'
/home/mjordan/projects/pjproject/pjmedia/lib/libpjmedia-videodev.so: undefined reference to `pjmedia_get_video_format_info'
```

Workaround

Run `configure` with either or both `--disable-video` or `--disable-v4l2`

Issue

After building `pipproject`, the dump provided by `ldconfig -p` doesn't display any libraries.

Workaround

Run `ldconfig` to re-configure dynamic linker run-time bindings

Uninstalling a Previous Version of `pipproject`

Typically, other versions of `pipproject` will be installed as static libraries. These libraries are not compatible with Asterisk and can confuse the build process for Asterisk 12. As such, any static libraries must be removed prior to installing the compatible version of `pipproject`.

`pipproject` provides an `uninstall` make target that will remove previous installations. It can be called from the `pipproject` source directory like:

```
# make uninstall
```

If you don't have an "uninstall" make target, you may need to fetch and merge the latest `pipproject` from <https://github.com/asterisk/pjproject>

Alternatively, the following should also remove all previously installed static libraries:

```
# rm -f /usr/lib/libpj*.a /usr/lib/libmilenage*.a /usr/lib/pkgconfig/libpjproject.pc
```

Configuring res_pjsip

- Overview
- Before You Configure
- Quick Start
- pjsip.conf Explained
 - Configuration Section Format
 - Section Types
 - **ENDPOINT** (res_pjsip)
 - **TRANSPORT** (res_pjsip)
 - **AUTH** (res_pjsip)
 - **AOR** (res_pjsip)
 - **REGISTRATION** (res_pjsip_outbound_registration)
 - **DOMAIN_ALIAS** (res_pjsip)
 - **ACL** (res_pjsip_acl)
 - **IDENTIFY** (res_pjsip_endpoint_identifier_ip)
 - **CONTACT** (res_pjsip)
 - Config Section Help and Defaults
 - Relationships of Configuration Objects in pjsip.conf
- Full res_pjsip configuration examples by scenario
 - An endpoint with a single SIP phone with inbound registration to Asterisk
 - A SIP trunk to your service provider, including outbound registration
 - Multiple endpoints with phones registering to Asterisk, using templates
- Dialing using chan_pjsip
 - Dialing from dialplan in extensions.conf
- Old to New - sip.conf to pjsip.conf example comparison
 - Example Endpoint Configuration
 - Example SIP Trunk Configuration
- Disabling res_pjsip and chan_pjsip

Overview

This page is intended to help an Asterisk administrator understand configuration fundamentals for the new SIP resources and channel driver included with Asterisk 12. It covers an explanation of configuration for pjsip.conf which configures the SIP resource modules utilized by the chan_pjsip driver.

If you are looking for info on how to configure sip.conf (the config used by the older SIP channel driver for Asterisk), then you'll want to go here [Creating SIP Accounts](#) for some basic info or check out the sip.conf sample file included in the /configs directory of your Asterisk source files.

Before You Configure

This page assumes certain knowledge, or that you have completed a few prerequisites

- You have [installed pjproject](#), a dependency for res_pjsip.
- You have Installed Asterisk including the [res_pjsip and chan_pjsip modules](#) (implying you installed their dependencies as well)
- You understand basic Asterisk concepts. Including the [role of extensions.conf](#) (dialplan) in your overall Asterisk configuration.

If you don't know anything about Asterisk yet, then you should probably start at the [Getting Started](#) section.

Quick Start

If you like to play before you read or figure out things as you go; here's a few quick steps to get you started.

- Understand that res_pjsip is configured through pjsip.conf. This is where you'll be configuring everything related to your SIP trunks or phones.
- Look at the [Full res_pjsip configuration examples by scenario](#) section. Grab the example most appropriate to your goal and use that to replace your pjsip.conf.
- Reference documentation for all configuration parameters is available on the wiki:
 - [Core res_pjsip configuration options](#)
 - [Configuration options for ACLs in res_pjsip_acl](#)
 - [Configuration options for outbound registration, provided by res_pjsip_outbound_registration](#)

- [Configuration options for endpoint identification by IP address, provided by res_pjsip_endpoint_identifier_ip](#)
- You'll need to tweak details in pjsip.conf and on your SIP device (for example IP addresses and authentication credentials) to get it working with Asterisk.
Refer back to this page, the sample pjsip.conf, or the reference documentation if you get confused.

pjsip.conf Explained

Configuration Section Format

pjsip.conf is a flat text file composed of **sections** like most configuration files used with Asterisk. Each **section** defines configuration for a **configuration object** within res_pjsip or an associated module.

Sections are identified by **names in square brackets**. (see [SectionName](#) below)

Each section has one or more **configuration options** that can be assigned a value by using an **equal sign** followed by a value. (see [ConfigOption](#) and [Value](#) below) These options and values are the configuration for a particular component of functionality provided by the configuration object's respective Asterisk modules.

Every section will have a **type** option that defines what kind of section is being configured. You'll see that in every example config section below.

Syntax for res_sip config objects

```
[ SectionName ]
ConfigOption = Value
ConfigOption = Value
```

Section Types

Below is a brief description of each section type and an example showing configuration of that section only. The module providing the configuration object related to the section is listed in parentheses next to each section name.

There are dozens of config options for some of the sections, but the examples below are very minimal for the sake of simplicity.



Option Values and Defaults

How do I know what values I can use for an option? Use the built-in configuration help at the CLI or view the wiki section listing all config option help text. You can use "config show help res_pjsip <configobject> <configoption>" to get help on a particular option. The output will typically describe the **default** value for an option as well. **[Link to list of config options goes here, once we have them pulled onto the wiki](#)**

Defaults: For many config options, it's very helpful to understand their default behavior. For example, endpoint's "transport=" option, if no value is assigned then Asterisk will *DEFAULT* to the first configured transport in pjsip.conf which is valid for the URI we are trying to contact.

ENDPOINT (res_pjsip)

Endpoint configuration provides numerous options relating to core SIP functionality and ties to other sections such as auth, aor and transport. You can't contact an endpoint without associating one or more AoR sections. An endpoint is essentially a profile for the configuration of a SIP endpoint such as a phone or remote server.

EXAMPLE BASIC CONFIGURATION

```
[6001]
type=endpoint
context=default
disallow=all
allow=ulaw
transport=simpletrans
auth=auth6001
aors=6001
```

If you want to define the Caller Id this endpoint should use, then add something like the following:

```
trust_id_outbound=yes
callerid=Spaceman Spiff <6001>
```

TRANSPORT (res_pjsip)

Configure how res_pjsip will operate at the transport layer. For example, it supports configuration options for protocols such as TCP, UDP or WebSockets and encryption methods like TLS/SSL. You can setup multiple transport sections and other sections (such as endpoints) could each use the same transport, or a unique one.



Reloading Config: Configuration for transport type sections can't be reloaded during run-time without a full module unload and load. You'll effectively need to restart Asterisk completely for your transport changes to take effect.

EXAMPLE BASIC CONFIGURATION

A basic UDP transport bound to all interfaces

```
[simpletrans]
type=transport
protocol=udp
bind=0.0.0.0
```

Or a TLS transport, with many possible options and parameters:

```
[simpletrans]
type=transport
protocol=tls
bind=0.0.0.0
;various TLS specific options below:
cert_file=
privkey_file=
ca_list_file=
cipher=
method=
```

AUTH (res_pjsip)

Authentication sections hold the options and credentials related to inbound or outbound authentication. You'll associate other sections such as endpoints or registrations to this one. Multiple endpoints or registrations can use a single auth config if needed.

EXAMPLE BASIC CONFIGURATION

An example with username and password authentication

```
[auth6001]
type=auth
auth_type=userpass
password=6001
username=6001
```

And then an example with MD5 authentication

```
[auth6001]
type=auth
auth_type=md5
md5_cred=51e63a3da6425a39aecc045ec45f1ae8
username=6001
```

AOR (res_pjsip)

A primary feature of AOR objects (Address of Record) is to tell Asterisk where an endpoint can be contacted. Without an associated AOR section, an endpoint cannot be contacted. AOR objects also store associations to mailboxes for MWI requests and other data that might relate to the whole group of contacts such as expiration and qualify settings.

When Asterisk receives an inbound registration, it'll look to match against available AORs.

Registrations: The name of the AOR section must match the user portion of the SIP URI in the "To:" header of the inbound SIP registration. That will usually be the "user name" set in your hard or soft phones configuration.

▼ EXAMPLE BASIC CONFIGURATION

First, we have a configuration where you are expecting the SIP User Agent (likely a phone) to register against the AOR. In this case, the contact objects will be created automatically. We limit the maximum contact creation to 1. We could do 10 if we wanted up to 10 SIP User Agents to be able to register against it.

```
[6001]
type=aor
max_contacts=1
```

Second, we have a configuration where you are **not** expecting the SIP User Agent to register against the AOR. In this case, you can assign contacts manually as follows. We don't have to worry about max_contacts since that option only affects the maximum allowed contacts to be created through external interaction, like registration.

```
[6001]
type=aor
contact=sip:6001@192.0.2.1:5060
```

Third, it's useful to note that you could define only the domain and omit the user portion of the SIP URI if you wanted. Then you could define the **user** portion dynamically in your dialplan when calling the Dial application. You'll likely do this when building an AOR/Endpoint combo to use for dialing out to an ITSP. For example: "Dial(PJSIP/\${EXTEN}@mytrunk)"

```
[mytrunk]
type=aor
contact=sip:203.0.113.1:5060
```

REGISTRATION (res_pjsip_outbound_registration)

The registration section contains information about an outbound registration. You'll use this when setting up a registration to another system whether it's local or a trunk from your ITSP.

▼ EXAMPLE BASIC CONFIGURATION

This example shows you how you might configure registration and outbound authentication against another Asterisk system, where the other system is using the older chan_sip peer setup.

This example is just the registration itself. You'll of course need the associated transport and auth sections. Plus, if you want to receive calls from the far end (who now knows where to send calls, thanks to your registration!) then you'll need endpoint, AOR and possibly identify sections setup to match inbound calls to a context in your dialplan.

```
[mytrunk]
type=registration
transport=simpletrans
outbound_auth=mytrunk
server_uri=sip:myaccountname@203.0.113.1:5060
client_uri=sip:myaccountname@192.0.2.1:5060
retry_interval=60
```

And an example that may work with a SIP trunking provider

```
[mytrunk]
type=registration
transport=simpletrans
outbound_auth=mytrunk
server_uri=sip:sip.example.com
client_uri=sip:1234567890@sip.example.com
retry_interval=60
```

DOMAIN_ALIAS (res_pjsip)

Allows you to specify an alias for a domain. If the domain on a session is not found to match an AoR then this object is used to see if we have an alias for the AoR to which the endpoint is binding. This sections name as defined in configuration should be the domain alias and a config option (domain=) is provided to specify the domain to be aliased.

▼ EXAMPLE BASIC CONFIGURATION

```
[example2.com]
type=domain_alias
domain=example.com
```

ACL (res_pjsip_acl)

The ACL module used by 'res_pjsip'. This module is independent of 'endpoints' and operates on all inbound SIP communication using res_pjsip. Features such as an Access Control List, as defined in the configuration section itself, or as defined in **acl.conf**. ACL's can be defined specifically for source IP addresses, or IP addresses within the contact header of SIP traffic.

▼ EXAMPLE BASIC CONFIGURATION

A configuration pulling from the acl.conf file:

```
[acl]
type=acl
acl=example_named_acl1
```

A configuration defined in the object itself:

```
[acl]
type=acl
deny=0.0.0.0/0.0.0.0
permit=209.16.236.0
permit=209.16.236.1
```

A configuration where we are restricting based on contact headers instead of IP addresses.

```
[acl]
type=acl
contactdeny=0.0.0.0/0.0.0.0
contactpermit=209.16.236.0
contactpermit=209.16.236.1
```

All of these configurations can be combined.

IDENTIFY (res_pjsip_endpoint_identifier_ip)

Controls how the res_pjsip_endpoint_identifier_ip module determines what endpoint an incoming packet is from. If you don't have an identify section defined, or else you have res_pjsip_endpoint_identifier_ip loading **after** res_pjsip_endpoint_identifier_user, then res_pjsip_endpoint_identifier_user will identify inbound traffic by pulling the user from the "From:" SIP header in the packet. Basically the module load order, and your configuration will both determine whether you identify by IP or by user.

▼ EXAMPLE BASIC CONFIGURATION

Its use is quite straightforward. With this configuration if Asterisk sees inbound traffic from 203.0.113.1 then it will match that to Endpoint 6001.

```
[6001]
type=identify
endpoint=6001
match=203.0.113.1
```

CONTACT (res_pjsip)

The contact config object effectively acts as an alias for a SIP URIs and holds information about an inbound registrations. Contact objects can be associated with an individual SIP User Agent and contain a few config options related to the connection. Contacts are created automatically upon registration to an AOR, or can be created manually by using the "contact=" config option in an AOR section. Manually configuring a CONTACT config object itself is outside the scope of this "getting started" style document.

Config Section Help and Defaults

Once we have the XML configuration help pulled onto the wiki we'll put a link here to that wiki section.

In the meantime use the built-in configuration help to your advantage. You can use "config show help res_pjsip <configobject> <configoption>" to get help on a particular option. That help will typically describe the default value for an option as well.

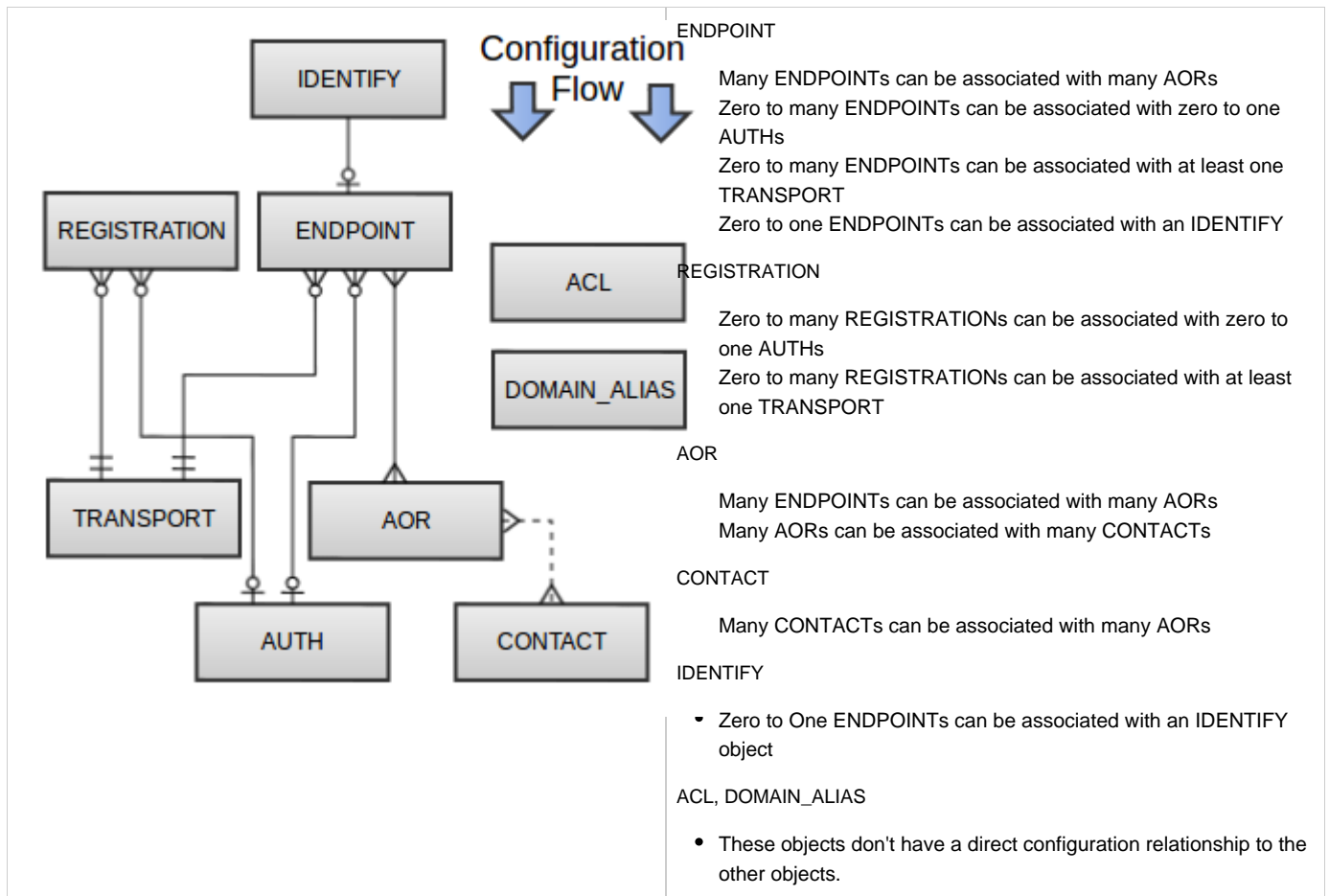
Relationships of Configuration Objects in pjsip.conf

Now that you understand the various configuration sections related to each config object, lets look at how they interrelate.

You'll see that the new SIP implementation within Asterisk is extremely flexible due to its modular design. A diagram will help you to visualize the relationships between the various configuration objects. The following entity relationship diagram covers only the configuration relationships between the objects. For example if an **endpoint** object requires authorization for registration of a SIP device, then you may associate a single **auth** object with the endpoint object. Though many endpoints could use the same or different auth objects.

Configuration Flow: This lets you know which direction the objects are associated to other objects. e.g. The identify config section has an option "endpoint=" which allows you to associate it with an endpoint object.

Entity Relationships	Relationship Descriptions
----------------------	---------------------------



Full res_pjsip configuration examples by scenario

Below are some sample configurations to demonstrate various scenarios with complete pjsip.conf files.

An endpoint with a single SIP phone with inbound registration to Asterisk

✓ [EXAMPLE CONFIGURATION](#)

```

;=====TRANSPORT

[simpletrans]
type=transport
protocol=udp
bind=0.0.0.0

;=====EXTENSION 6001

[6001]
type=endpoint
context=internal
disallow=all
allow=ulaw
transport=simpletrans
auth=auth6001
aors=6001

[auth6001]
type=auth
auth_type=userpass
password=6001
username=6001

[6001]
type=aor
max_contacts=1

```

- `auth=` is used for the endpoint as opposed to `outbound_auth=` since we want to allow inbound registration for this endpoint
- `max_contacts=` is set to something non-zero as we want to allow contacts to be created through registration

A SIP trunk to your service provider, including outbound registration

▼ EXAMPLE CONFIGURATION

Trunks are a little tricky since many providers have unique requirements. Your final configuration may differ from what you see here.

```

;=====TRANSPORTS

[simpletrans]
type=transport
protocol=udp
bind=0.0.0.0

;=====TRUNK

[mytrunk]
type=registration
transport=simpletrans
outbound_auth=mytrunk
server_uri=sip:sip.example.com
client_uri=sip:1234567890@sip.example.com
retry_interval=60

[mytrunk]
type=auth
auth_type=userpass
password=1234567890
username=1234567890

[mytrunk]
type=aor
contact=sip:203.0.113.1:5060

[mytrunk]
type=endpoint
transport=simpletrans
context=from-external
disallow=all
allow=ulaw
outbound_auth=mytrunk
aors=mytrunk

[mytrunk]
type=identify
endpoint=mytrunk
match=203.0.113.1

```

- "contact=sip:203.0.113.1:5060", we don't define the user portion statically since we'll set that dynamically in dialplan when we call the Dial application. See the dialing examples in the section "Dialing using chan_pjsip" for more.
- "outbound_auth=mytrunk", we use "outbound_auth" instead of "auth" since the provider isn't typically going to authenticate with us when calling, but we will probably have to authenticate when calling through them.
- We use an identify object to map all traffic from the provider's IP as traffic to that endpoint since the user portion of their From: header may vary with each call.
- This example assumes that sip.example.com resolves to 203.0.113.1

Multiple endpoints with phones registering to Asterisk, using templates

✓ EXAMPLE CONFIGURATION

We want to show here that generally, with a large configuration you'll end up using templates to make configuration easier to handle when scaling. This avoids having redundant code in every similar section that you create.

```

;=====TRANSPORT

```



```
[simpletrans]
type=transport
protocol=udp
bind=0.0.0.0

;=====ENDPOINT TEMPLATES

[endpoint-basic]()
type=endpoint
transport=simpletrans
context=internal
disallow=all
allow=ulaw

[auth-userpass]()
type=auth
auth_type=userpass

[aor-single-reg]()
type=aor
max_contacts=1

;=====EXTENSION 6001

[6001](endpoint-basic)
auth=auth6001
aors=6001

[auth6001](auth-userpass)
password=6001
username=6001

[6001](aor-single-reg)

;=====EXTENSION 6002

[6002](endpoint-basic)
auth=auth6002
aors=6002

[auth6002](auth-userpass)
password=6002
username=6002

[6002](aor-single-reg)

;=====EXTENSION 6003

[6003](endpoint-basic)
auth=auth6003
aors=6003

[auth6003](auth-userpass)
password=6003
```

```
username=6003

[6003](aor-single-reg)
```

Obviously the larger your configuration is, the more templates will benefit you. Here we just break apart the endpoints with templates, but you could do that with any config section that needs instances with variation, but where each may share common settings with their peers.

Dialing using chan_pjsip

Dialing from dialplan in extensions.conf

We are assuming you already know a little bit about the Dial application here. To see the full help for it, see "core show help application dial" on the Asterisk CLI, or see [Application_Dial](#)

Below we'll simply dial an endpoint using the chan_pjsip channel driver. This is really going to look at the AOR of the same name as the endpoint and start dialing the contacts associated.

```
exten => _6XXX,1,Dial(PJSIP/${EXTEN})
```

Here's how you would dial with an explicit SIP URI, user and domain, via an endpoint (in this case dialing out a trunk), but not using its associated AOR/contact objects.

```
exten => _9NXXNXXXXXX,1,Dial(PJSIP/mytrunk/sip:${EXTEN:1}@203.0.113.1:5060)
```

This uses a contact (and its domain) set in the AOR associated with the **mytrunk** endpoint, but still explicitly sets the user portion of the URI in the dial string. For the AOR's contact, you would define it in the AOR config without the user name.

```
exten => _9NXXNXXXXXX,1,Dial(PJSIP/${EXTEN:1}@mytrunk)
```

Old to New - sip.conf to pjsip.conf example comparison

We want to provide some examples of what similar configurations would look like between the old sip.conf and the new pjsip.conf

These examples contain only the configuration required for sip.conf/pjsip.conf as the configuration for other files should be the same excepting the Dial statements in your extensions.conf

There is also a script available to provide a basic conversion of a sip.conf config to a pjsip.conf config. **ADD A LINK TO SIP.CONF to PJSIP.CONF SCRIPT WHEN READY**

Example Endpoint Configuration

This examples shows the configuration required for:

- two SIP phones need to make calls to or through Asterisk, we also want to be able to call them from Asterisk
- for them to be identified as users (in the old chan_sip) or endpoints (in the new res_sip/chan_pjsip)
- both devices need to use username and password authentication
- 6001 is setup to allow registration to Asterisk, and 6002 is setup with a static host/contact

sip.conf	pjsip.conf
----------	------------

```
[general]
udpbindaddr=0.0.0.0
```

```
[6001]
type=friend
host=dynamic
disallow=all
allow=ulaw
context=internal
secret=1234
```

```
[6002]
type=friend
host=192.0.2.1
disallow=all
allow=ulaw
context=internal
secret=1234
```

```
[simpletrans]
type=transport
protocol=udp
bind=0.0.0.0
```

```
[6001]
type = endpoint
transport = simpletrans
context = internal
disallow = all
allow = ulaw
aors = 6001
auth = auth6001
```

```
[6001]
type = aor
max_contacts = 1
```

```
[auth6001]
type=auth
auth_type=userpass
password=1234
username=6001
```

```
[6002]
type = endpoint
transport = simpletrans
context = internal
disallow = all
allow = ulaw
aors = 6002
auth = auth6002
```

```
[6002]
type = aor
contact = sip:6002@192.0.2.1:5060
```

```
[auth6002]
type=auth
auth_type=userpass
password=1234
username=6001
```

Example SIP Trunk Configuration

This shows configuration for a SIP trunk as would typically be provided by an ITSP. That is registration to a remote server, authentication to it and a peer/endpoint setup to allow inbound calls from the provider.

- SIP provider requires registration to their server with a username of "myaccountname" and a password of "1234567890"
- SIP provider requires registration to their server at the address of 203.0.113.1:5060
- SIP provider requires outbound calls to their server at the same address of registration, plus using same authentication details.
- SIP provider will call your server with a user name of "mytrunk". Their traffic will only be coming from 203.0.113.1

sip.conf

pjsip.conf

```
[general]
udpbindaddr=0.0.0.0

register =>
myaccountname:1234567890@203.0.113.1:5060
0

[mytrunk]
type=friend
secret=1234567890
username=myaccountname
host=203.0.113.1
disallow=all
allow=ulaw
context=from-external
```

```
[simpletrunk]
type=transport
protocol=udp
bind=0.0.0.0

[mytrunk]
type=registration
transport=simpletrunk
outbound_auth=mytrunk
server_uri=sip:myaccountname@203.0.113.1:5060
client_uri=sip:myaccountname@192.0.2.1:5060

[mytrunk]
type=auth
auth_type=userpass
password=1234567890
username=myaccountname

[mytrunk]
type=aor
contact=sip:203.0.113.1:5060

[mytrunk]
type=endpoint
transport=simpletrunk
context=from-external
disallow=all
allow=ulaw
outbound_auth=mytrunk
aors=mytrunk

[mytrunk]
type=identify
endpoint=mytrunk
match=203.0.113.1
```

Disabling res_pjsip and chan_pjsip

There are several methods to disable or remove modules in Asterisk. Which method is best depends on your intent.

If you have built Asterisk with the PJSIP modules, but don't intend to use them at this moment, you might consider the following:

1. Edit the file **modules.conf** in your Asterisk configuration directory. (typically /etc/asterisk/)

```
noload => res_pjsip_pubsub.so
noload => res_pjsip_session.so
noload => chan_pjsip.so
noload => res_pjsip_exten_state.so
noload => res_pjsip_log_forwarder.so
```

Having a noload for the above modules should (at the moment of writing this) prevent any PJSIP related modules from loading.

2. Restart Asterisk!

Other possibilities would be:

- Remove all PJSIP modules from the modules directory (often, `/usr/lib/asterisk/modules`)
- Remove the configuration file (`pjsip.conf`)
- Un-install and re-install Asterisk with no PJSIP related modules.
- If you are wanting to use `chan_pjsip` alongside `chan_sip`, you could change the port or bind interface of your `chan_pjsip` transport in `pjsip.conf`

Configuring res_pjsip to work through NAT

Here we can show some examples of working configuration for Asterisk's SIP channel driver when Asterisk is behind NAT (Network Address Translation).

Asterisk and Phones Connecting Through NAT to an ITSP

This example should apply for most simple NAT scenarios that meet the following criteria:

- Asterisk and the phones are on a private network.
- There is a router interfacing the private and public networks. Where the public network is the Internet.
- The router is performing Network Address Translation and Firewall functions.
- The router is configured for port-forwarding, where it is mapping the necessary ranges of SIP and RTP traffic to your internal Asterisk server.

In this example the router is port-forwarding WAN inbound TCP/UDP 5060 and UDP 10000-20000 to LAN 192.0.2.10

This example was based on a configuration for the ITSP [SIP.US](#) and assuming you swap out the addresses and credentials for real ones, it should work for a SIP.US SIP account.

Devices Involved in the Example

Using [RFC5737](#) documentation addresses

Device	IP in example
VOIP Phone(6001)	192.0.2.20
PC/Asterisk	192.0.2.10
Router	LAN: 192.0.2.1 WAN: 198.51.100.5
ITSP SIP gateway	203.0.113.1(gw1.example.com) 203.0.113.2(gw2.example.com)

For the sake of a complete example and clarity, in this example we use the following fake details:

ITSP Account number: 1112223333

DID number provided by ITSP: 19998887777

pjsip.conf Configuration

We are assuming you have already read the Configuring res_pjsip page and have a basic understanding of Asterisk. For this NAT example, the important config options to note are **local_net**, **external_media_address** and **external_signaling_address** in the transport type section and **direct_media** in the endpoint section. The rest of the options may depend on your particular configuration, phone model, network settings, ITSP, etc. The key is to make sure you have those three options set appropriately.

local_net

This is the IP network that we want to consider our local network. For communication to addresses within this range, we won't apply any NAT-related settings, such as the external* options below.

external_media_address

This is the external IP address to use in RTP handling. When a request or response is sent out from Asterisk, if the destination of the message is outside the IP network defined in the option 'local_net', and the media address in the SDP is within the localnet network, then the media address in the SDP will be rewritten to the value defined for 'external_media_address'.

external_signaling_address

This is much like the external_media_address setting, but for SIP signaling instead of RTP media. The two external* options mentioned here should be set to the same address unless you separate your signaling and media to different addresses or servers.

direct_media

Determines whether media may flow directly between endpoints

Together these options make sure the far end knows where to send back SIP and RTP packets, and `direct_media` ensures Asterisk stays in the media path. This is important, because our Asterisk system has a private IP address that the ITSP cannot route to. We want to make sure the SIP and RTP traffic comes back to the WAN/Public internet address of our router. The sections prefixed with "sipus" are all configuration needed for inbound and outbound connectivity of the SIP trunk, and the sections named 6001 are all for the VOIP phone.

```
[transport-udp-nat]
type=transport
protocol=udp
bind=0.0.0.0
local_net=192.0.2.0/24
local_net=127.0.0.1/32
external_media_address=198.51.100.5
external_signaling_address=198.51.100.5

[sipus_reg]
type=registration
transport=transport-udp-nat
outbound_auth=sipus_auth
server_uri=sip:gw1.example.com
client_uri=sip:1112223333@gw1.example.com
contact_user=19998887777
retry_interval=60

[sipus_auth]
type=auth
auth_type=userpass
password=*****
username=1112223333
realm=gw1.example.com

[sipus_endpoint]
type=endpoint
transport=transport-udp-nat
section=from-external
disallow=all
allow=ulaw
outbound_auth=sipus_auth
aors=sipus_aor
direct_media=no
from_domain=gw1.example.com

[sipus_aor]
type=aor
contact=sip:gw1.example.com
contact=sip:gw2.example.com

[sipus_identify]
type=identify
endpoint=sipus_endpoint
match=203.0.113.1
match=203.0.113.2

[6001]
type=endpoint
section=from-internal
```

```
disallow=all  
allow=ulaw  
transport=transport-udp-nat  
auth=6001  
aors=6001  
direct_media=no
```

```
[6001]  
type=auth  
auth_type=userpass  
password=*****  
username=6001
```



```
[6001]  
type=aor  
max_contacts=2
```

For Remote Phones Behind NAT

In the above example we assumed the phone was on the same local network as Asterisk. Now, perhaps Asterisk is exposed on a public address, and instead your phones are remote and behind NAT, or maybe you have a double NAT scenario?

In these cases you will want to consider the below settings for the remote endpoints.

media_address

IP address used in SDP for media handling

At the time of SDP creation, the IP address defined here will be used as the media address for individual streams in the SDP.

NOTE: Be aware that the 'external_media_address' option, set in Transport configuration, can also affect the final media address used in the SDP.

rtp_symmetric

Enforce that RTP must be symmetric. Send RTP back to the same port we received it from.

force_rport

Force RFC3581 compliant behavior even when no rport parameter exists. Basically always send SIP responses back to the same port we received SIP requests from.

direct_media

Determines whether media may flow directly between endpoints.

Clients Supporting ICE,STUN,TURN

This is really relevant to media, so look to the [section here](#) for basic information on enabling this support and we'll add relevant examples later.

Setting up PJSIP Realtime

- Overview
 - Installing Dependencies
 - Creating the MySQL Database
 - Installing and Using Alembic
 - Configuring ODBC
 - Connecting PJSIP Sorcery to the Realtime Database
 - Realtime Configuration
 - Asterisk Startup Configuration
 - Asterisk PJSIP configuration
 - Endpoint Population
 - A Little Dialplan
 - Conclusion

Overview

This tutorial describes the configuration of Asterisk's PJSIP channel driver with the "realtime" database storage backend. The realtime interface allows storing much of the configuration of PJSIP, such as endpoints, auths, aors and more, in a database, as opposed to the normal flat-file storage of `pjsip.conf`.

Installing Dependencies

For the purposes of this tutorial, we will assume a base Ubuntu 12.0.4.3 x86_64 server installation, with the OpenSSH server and LAMP server options, and that Asterisk will use its ODBC connector to reach a back-end MySQL database.

Beyond the normal packages needed to install Asterisk 12 on such a server (build-essential, libncurses5-dev, uuid-dev, libjansson-dev, libxml2-dev, libsqlite3-dev) as well as the [Installation of pjproject](#), you will need to install the following packages:

- unixodbc and unixodbc-dev
 - ODBC and the development packages for building against ODBC
- libmyodbc
 - The ODBC to MySQL interface package
- python-dev and python-pip
 - The Python development package and the pip package to allow installation of Alembic
- python-mysqldb
 - The Python interface to MySQL, which will be used by Alembic to generate the database tables

So, from the CLI, perform:

```
# apt-get install unixodbc unixodbc-dev libmyodbc python-dev python-pip python-mysqldb
```

Once these packages are installed, check your Asterisk installation's **make menuconfig** tool to make sure that the **res_config_odbc** and **res_odbc** resource modules, as well as the **res_pjsip_XXX** modules are selected for installation. If they are, then go through the normal Asterisk installation process: **.configure; make; make install**

And, if this is your first installation of Asterisk, be sure to install the sample files: **make samples**

Creating the MySQL Database

Use the **mysqladmin** tool to create the database that we'll use to store the configuration. From the Linux CLI, perform:

```
# mysqladmin -u root -p create asterisk
```

This will prompt you for your MySQL database password and then create a database named **asterisk** that we'll use to store our PJSIP configuration.

Installing and Using Alembic

Alembic is a full database migration tool, with support for upgrading the schemas of existing databases, versioning of schemas, creation of new tables and databases, and a whole lot more. A good guide on using Alembic with Asterisk can be found on the [Managing Realtime Databases with Alembic](#) wiki page. A shorter discussion of the steps necessary to prep your database will follow.

First, install Alembic:

```
# pip install alembic
```

Then, move to the Asterisk source directory containing the Alembic scripts:

```
# cd contrib/ast-db-manage/
```

Next, edit the **config.ini.sample** file and change the **sqlalchemy.url** option, e.g.

```
sqlalchemy.url = mysql://root:password@localhost/asterisk
```

such that the URL matches the username and password required to access your database.

Then rename the config.ini.sample file to config.ini

```
# cp config.ini.sample config.ini
```

Finally, use Alembic to setup the database tables:

```
# alembic -c config.ini upgrade head
```

You'll see something similar to:

```
# alembic -c config.ini upgrade head
INFO [alembic.migration] Context impl MySQLImpl.
INFO [alembic.migration] Will assume non-transactional DDL.
INFO [alembic.migration] Running upgrade None -> 4da0c5f79a9c, Create tables
INFO [alembic.migration] Running upgrade 4da0c5f79a9c -> 43956d550a44, Add tables for pjsip
#
```

You can then connect to MySQL to see that the tables were created:

```
# mysql -u root -p -D asterisk

mysql> show tables;
+-----+
| Tables_in_asterisk |
+-----+
| alembic_version    |
| iaxfriends         |
| meetme             |
| musiconhold        |
| ps_aors             |
| ps_auths           |
| ps_contacts        |
| ps_domain_aliases  |
| ps_endpoint_id_ips |
| ps_endpoints       |
| sippeers           |
| voicemail          |
+-----+
12 rows in set (0.00 sec)
mysql> quit
```

Configuring ODBC

Now that we have our MySQL database created and populated, we'll need to setup ODBC and Asterisk's ODBC resource to access the database. First, we'll tell ODBC how to connect to MySQL. To do this, we'll edit the **/etc/odbcinst.ini** configuration file. Your file should look something like:

/etc/odbcinst.ini

```
[MySQL]
Description = ODBC for MySQL
Driver = /usr/lib/x86_64-linux-gnu/odbc/libmyodbc.so
Setup = /usr/lib/x86_64-linux-gnu/odbc/libodbcmyS.so
UsageCount = 2
```

Next, we'll tell ODBC **which** MySQL database to use. To do this, we'll edit the **/etc/odbc.ini** configuration file and create a database handle called **asterisk**. Your file should look something like:

/etc/odbc.ini

```
[asterisk]
Driver = MySQL
Description = MySQL connection to 'asterisk' database
Server = localhost
Port = 3306
Database = asterisk
UserName = root
Password = password
Socket = /var/run/mysqld/mysqld.sock
```

Take care to use your database access UserName and Password, and not necessarily what's defined in this example.

Now, we need to configure Asterisk's ODBC resource, `res_odbc`, to connect to the ODBC **asterisk** database handle that we just created. `res_odbc` is configured using the `/etc/asterisk/res_odbc.conf` configuration file. There, you'll want:

/etc/asterisk/res_odbc.conf

```
[asterisk]
enabled => yes
dsn => asterisk
username => root
password => password
pre-connect => yes
```

Again, take care to use the proper username and password.

Now, you can start Asterisk and you can check its connection to your "asterisk" MySQL database using the "asterisk" `res_odbc` connector to ODBC. You can do this by executing "odbc show" from the Asterisk CLI. If everything went well, you'll see:

```
# asterisk -vvvvv
*CLI> odbc show

ODBC DSN Settings
-----
Name: asterisk
DSN: asterisk
Last connection attempt: 1969-12-31 18:00:00
Pooled: No
Connected: Yes
*CLI>
```

Connecting PJSIP Sorcery to the Realtime Database

The PJSIP stack uses a new data abstraction layer in Asterisk called **sorcery**. Sorcery lets a user build a hierarchical layer of data sources for Asterisk to use when it retrieves, updates, creates, or destroys data that it interacts with. This tutorial focuses on getting PJSIP's configuration stored in a realtime back-end; the rest of the details of sorcery are beyond the scope of this page.

PJSIP bases its configuration on types of objects. For more information about these types of objects, please refer to the [Configuring res_pjsip](#) wiki page. In this case, we have a total of five objects we need to configure in Sorcery:

- endpoint
- auth
- aor
- domain
- identify

Sorcery is configured using the `/etc/asterisk/sorcery.conf` configuration file. So, we need to add the following lines to the file:

/etc/asterisk/sorcery.conf

```
[res_pjsip] ; Realtime PJSIP configuration wizard
endpoint=realtime,ps_endpoints
auth=realtime,ps_auths
aor=realtime,ps_aors
domain_alias=realtime,ps_domain_aliases
identify=realtime,ps_endpoint_id_ips
```

The items use the following nomenclature:

```
{object_type} = {sorcery_wizard_name},{wizard_arguments}
```

In our case, the `sorcery_wizard_name` is **realtime**, and the **wizard_arguments** are the name of the database connector ("asterisk") to associate with our object types.

Realtime Configuration

Since we've associated the PJSIP objects with database connector types, we now need to tell Asterisk to use a database backend with the object types, and not just the flat pjsip.conf file. To do this, we modify the `/etc/asterisk/extconfig.conf` configuration file to provide these connections.

Open `extconfig.conf` (`/etc/asterisk/extconfig.conf`) and add the following lines to the 'settings' configuration section

/etc/asterisk/extconfig.conf

```
ps_endpoints => odbc,asterisk
ps_auths => odbc,asterisk
ps_aors => odbc,asterisk
ps_domain_aliases => odbc,asterisk
ps_endpoint_id_ips => odbc,asterisk
```

At this point, Asterisk is nearly ready to use the tables created by alembic with PJSIP to configure endpoints, authorization, AORs, domain aliases, and endpoint identifiers.



A warning for adventurous types:

Sorcery.conf allows you to try to configure other PJSIP objects such as transport using realtime and it currently won't stop you from doing so. However, some of these object types should not be used with realtime and this can lead to errant behavior.

Asterisk Startup Configuration

Now, we need to configure Asterisk to load its ODBC driver at an early stage of startup, so that it's available when any other modules might need to take advantage of it. Also, we're going to prevent the old `chan_sip` channel driver from loading, since we're only worried about PJSIP.

To do this, edit the `/etc/asterisk/modules.conf` configuration file. In the **[modules]** section, add the following lines:

/etc/asterisk/modules.conf

```
preload => res_odbc.so
preload => res_config_odbc.so
noload => chan_sip.so
```

Asterisk PJSIP configuration

Next, we need to configure a transport in `/etc/asterisk/pjsip.conf`. PJSIP transport object types are not stored in realtime as unexpected results can occur. So, edit it and add the following lines:

/etc/asterisk/pjsip.conf

```
[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0
```

Here, we created a transport called **transport-udp** that we'll reference in the next section.

Endpoint Population

Now, we need to create our endpoints inside of the database. For this example, we'll create two peers, 101 and 102, that register using the totally insecure passwords "101" and "102" respectively. Here, we'll be populating data directly into the database using the MySQL interactive tool.

```
# mysql -u root -p -D asterisk;
mysql> insert into ps_aors (id, max_contacts) values (101, 1);
mysql> insert into ps_aors (id, max_contacts) values (102, 1);
mysql> insert into ps_auths (id, auth_type, password, username) values (101, 'userpass', 101, 101);
mysql> insert into ps_auths (id, auth_type, password, username) values (102, 'userpass', 102, 102);
mysql> insert into ps_endpoints (id, transport, aors, auth, context, disallow, allow, direct_media) values (101, 'transport-udp',
'101', '101', 'testing', 'all', 'g722', 'no');
mysql> insert into ps_endpoints (id, transport, aors, auth, context, disallow, allow, direct_media) values (102, 'transport-udp',
'102', '102', 'testing', 'all', 'g722', 'no');
mysql> quit;
```

In this example, we first created an **aor** for each peer, one called **101** and the other **102**.

Next, we created an **auth** for each peer with a userpath of **101** and **102**.

Then, we created two endpoints, **101** and **102**, each referencing the appropriate **auth** and **aor**, and we elected for the G.722 codec and we forced media to route inside of Asterisk (not the default behavior of Asterisk).

Now, you can start Asterisk and you can check to see if it's finding your PJSIP endpoints in the database. You can do this by executing "pjsip show endpoints" from the Asterisk CLI. If everything went well, you'll see:

```
# asterisk -vvvvv
*CLI> pjsip show endpoints
Endpoints:
101
102
*CLI>
```

A Little Dialplan

Now that we have our PJSIP endpoints stored in our MySQL database, let's add a little dialplan so that they can call each other. To do this, edit Asterisk's **/etc/asterisk/extensions.conf** file and add the following lines to the end:

/etc/asterisk/extensions.conf

```
[testing]
exten => _1XX,1,NoOp( )
same => n,Dial(PJSIP/${EXTEN})
```

Conclusion

Now, start Asterisk back up, or reload it using **core reload** from the Asterisk CLI, register your two SIP phones using the 101/101 and 102/102 credentials, and make a call.

Managing Realtime Databases with Alembic

Overview

Asterisk 12 now uses [Alembic](#) to help manage Asterisk Realtime Database schemas. This includes creation of SQL scripts for a variety of database vendors, but also much more. Alembic is a full database migration tool, with support for upgrading the schemas of existing databases, versioning of schemas, creation of new tables and databases, and a whole lot more. This page covers basic configuration of the Alembic configuration file for usage with Asterisk Realtime as well as basic usage of Alembic. While a full description of Alembic is beyond the scope of this page, the information on this page should help an Asterisk administrator create or upgrade an Asterisk installation.

Before you Begin

This tutorial assumes you already have some experience in setting up Realtime configuration with Asterisk for other modules. This page will not describe how to set up backend database connectors, and is written under the assumption that you will be using ODBC to connect to your database since the ODBC adaptor is capable of connecting to most commonly used database servers. For more information on configuring and setting up Asterisk Realtime, see [Asterisk Realtime Database](#) configuration.

Installing Alembic

If you don't already have Alembic installed, perform the following:



This does assume that you have pip installed. If you do not have pip installed, `easy_install` should work just as well. If you don't have pip or `easy_install` (or Python), then you should probably install those first.

```
$ pip install alembic
```

And that's it!

Building the Database Tables

Alembic scripts were added to Asterisk in Asterisk 12, and will allow you to automatically populate your database with tables for most of the commonly used configuration options. The scripts are located in the [Asterisk contrib/ast-db-manage](#) folder:

```
$ cd contrib/ast-db-manage
```

For the rest of this tutorial, we will assume that operations will be taken in the context of that directory.

Within this directory, you will find a configuration sample file, `config.ini.sample`, which will need to be edited to connect to your database of choice. Open this file in your text editor of choice and then save a copy of this sample file as `config.ini` - this will serve as the configuration file you actually use with Alembic.

There are two different parameters in `config.ini` that require review: `sqlalchemy.url` and `script_location`. The first specifies the database to upgrade; the second which upgrades to perform.

1. Update `sqlalchemy.url` to the URL for your database. An example is shown below for a MySQL database:

```
sqlalchemy.url = mysql://root:password@localhost/asterisk
```

This would connect to a MySQL database as user `root` with password `password`. The database is `asterisk`, located on `localhost`. Different databases will require different URL schemas; however, they should in general follow the format outlined above. Alembic supports many different database technologies, including `oracle`, `postgresql`, and `mssql`.

For more information, see the Alembic documentation on SQLAlchemy URLs: http://docs.sqlalchemy.org/en/rel_0_8/core/engines.html#database-urls

2. Update `script_location` to the schema to update. Asterisk currently supports two sets of schemas:
 - a. `config` - the set of schemas for Asterisk Realtime databases
 - b. `voicemail` - the schema for ODBC VoiceMail

**I'm sorry Dave, I'm afraid I can't let you do that.**

Using config.ini for Alembic will populate tables for all of the configuration objects that can be populated this way, so if you really don't want a table for sip peers, iax friends, voicemail, meetme, and music on hold, you may need to exercise a little fine control. Back up your database before continuing and be prepared to delete tables that you don't want when you are finished.

Your config.ini should be ready for use at this point, so close your text editor and return to the terminal. Then run:

```
$ alembic -c config.ini upgrade head
```

**Alembic makes upgrading less painful**

As Asterisk changes and new fields are made controllable via realtime, the Alembic scripts will be updated as well and you will be able to simply run the alembic upgrade command again in order to modify your database. Always exercise due diligence and backup your database before upgrading though. Tables can be fixed easily. Repopulating the data if it's lost however isn't.

At this point, if you configured your config.ini to connect to the database properly, your tables should be ready.

Getting Started with ARI

Asterisk 12 introduces the [Asterisk REST Interface](#), a set of RESTful API's for building Asterisk based applications. This article will walk you through getting ARI up and running.

Before you begin, you will need to [install Asterisk](#). This article also assumes that you are familiar with

You can find some historical documents [on the wiki](#) about the development and architecture of ARI.

- 1 [Overview](#)
- 2 [Configuration](#)
 - 2.1 [http.conf](#)
 - 2.2 [ari.conf](#)
 - 2.3 [extensions.conf](#)
 - 2.4 [Phone configuration](#)
- 3 [Authenticating Requests](#)
- 4 [Using Swagger-UI](#)
- 5 [Connecting the WebSocket](#)
- 6 [Doing stuff](#)
 - 6.1 [Manipulating the channel](#)
 - 6.2 [Asynchronous operations](#)
 - 6.3 [Bridging](#)
- 7 [Recommended Practices](#)
 - 7.1 [Don't access ARI directly from a web page](#)
 - 7.2 [Use an abstraction layer](#)
- 8 [WebSocket client libraries](#)

Overview

There are three main components to building an ARI application.

The first, obviously, is **the RESTful API** itself. The API is documented using [Swagger](#), a lightweight specification for documenting RESTful API's. The Swagger API docs are used to generate validations and boilerplate in Asterisk itself, along with [static wiki documentation](#), and interactive documentation using [Swagger-UI](#).

Then, Asterisk needs to send asynchronous events to the application (new channel, channel left a bridge, channel hung up, etc). This is done using a **WebSocket on /ari/events**. Events are sent as JSON messages, and are documented on the [REST Data Models page](#). (See the list of subtypes for the [Message data model](#).)

Finally, connecting the dialplan to your application is the [Stasis\(\) dialplan application](#). From within the dialplan, you can send a channel to `Stasis()`, specifying the name of the external application, along with optional arguments to pass along to the application.

Configuration

The sample configuration files, along with the `config show help` command, give full details of the Asterisk configuration files. Here's a sample set of file one might use to build a simple ARI application.

http.conf

In order to use ARI, Asterisk's HTTP server must be enabled. You should also consider the security of your interface when configuring the HTTP server.

http.conf

```
[general]
enabled=yes
bindaddr=127.0.0.1 ; For applications to run on a different machine, use
                    ; 0.0.0.0 or specify an address
;
; When running applications on a different machine, consider using TLS for
; your HTTP connections
;
;tlsenable=yes           ; enable tls - default no.
;tlsbindaddr=0.0.0.0:8089 ; address and port to bind to - default is
                        ; bindaddr and port 8089.
;tlscertfile=</path/to/certificate.pem> ; path to the certificate file
                                ; (*.pem) only.
;tlsprivatekey=</path/to/private.pem> ; path to private key file
                                ; (*.pem) only.
```

ari.conf

ARI itself is configured in the `ari.conf` file. For this example, we don't need pretty printing, so that's disabled.

We will be accessing the API from the Swagger-UI app hosted on ari.asterisk.org. This requires the `allowed_origins` option to be set, in order to access the API in a cross-origin sort of manner.

ari.conf

```
[general]
enabled=yes
;pretty=yes           ; we don't need pretty-printing of the JSON responses in this
                    ; example, but you might if you use curl a lot.
;
; In this example, we are going to use the version of Swagger-UI that is hosted
; at ari.asterisk.org. In order to get past CORS restrictions in the browser,
; That origin needs to be added to the allowed_origins list.
;
allowed_origins=ari.asterisk.org

[hey]
type=user
password=peekaboo
;read_only=no        ; Set to yes for read-only applications
;
; For the security conscious, you probably don't want to put plaintext passwords
; in the configuration file. ARI supports the use of crypt(3) for password
; storage. You can encrypt a password using the 'ari mkpasswd' command line
; command. Note that the protocols supported by crypt(3) are system specific,
; so check 'man 3 crypt' to see what's available on your system.
;
;password_format=crypt
;password=$6$u8RH5kQma8DV5$M0gydWzRGv/vuxtRXl306qpfi81Kr13F.QbqellcGARTFZ7GUWgERJ/OjD8cPA
;wItR/VMapo7bsHALqDPVnJX0
```

extensions.conf

If you would like to accept calls into your application, they need to be sent to the `Stasis()` application in the dialplan.

extensions.conf

```
[default]
exten => 7000,1,Noop()
    same => n,Stasis(hello,world) ; hello is the name of the application
                                   ; world is its argument list
    same => n,Hangup()
```

Phone configuration

Configuring phones is outside the scope of this walk through. I recommend configuring a SIP smart phone, simply because it makes things easier when dealing with multiple calls.

Authenticating Requests

ARI requests (both the RESTful API and the WebSocket) must be authenticated. Two authentication schemes are support.

1. `?api_key` query parameter. This is the default method supported by Swagger-UI
 - a. The `api_key` is "username:password"
2. HTTP Basic authentication

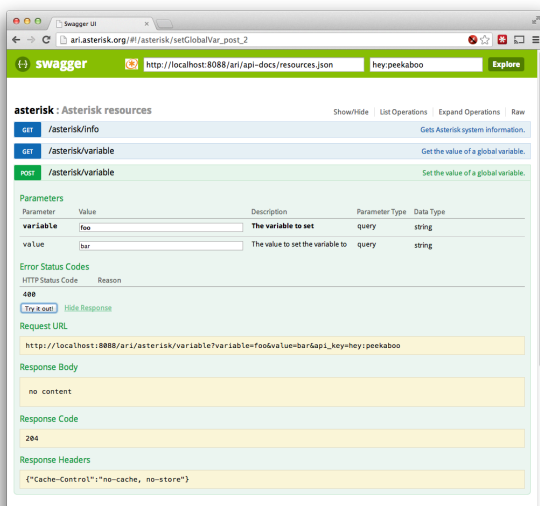
Using Swagger-UI

One of the advantages of documenting our RESTful API using Swagger is the ability to generate interactive documentation.

Swagger-UI is a pure HTML+JavaScript application which can download Swagger api-docs, and generate an interactive web page which allows you to view resources, their operations, and submit API requests directly from the documentation.

A fork of Swagger-UI is hosted on ari.asterisk.org, which enables DELETE operations (which are disabled by default in Swagger-UI), and sets the default URL to what it would be running Asterisk on your local system.

In order to access ARI, you have to populate the `api_key` field with a `username:password` configured in `ari.conf`. You should also set `allowed_origins` in `ari.conf` to allow the site hosting Swagger-UI to access ARI.



Connecting the WebSocket

When Asterisk is running, your application will need to connect to the WebSocket in order to receive events from Asterisk. There are several [WebSocket client libraries](#) available, covering most platforms and languages.

For this example, we will simply use the [wscat.py script](#). This script requires the [websocket-client library](#), so be sure to `pip install websocket-client` first.

```
$ wscat.py 'ws://localhost:8088/ari/events?app=hello&api_key=hey'
```

The WebSocket is now connected. As events are received, `wscat` will print them to stdout. Every received event will have:

- `type` - text string naming the event. This is the model id from the `/ari/api-docs/events.json` API doc.
- `application` - The name of the application receiving the event. In our example, the name is `hello`.
- `timestamp` - Time of the event. Most events have a `timestamp` field, but not all.

Doing stuff

Okay, first try something simple. Let's just dial the application. Use whatever telephony device you've configured with Asterisk to dial extension 7000. You should see this message come from the WebSocket:

"type": "StasisStart"

```
{
  "application": "hello",
  "args": [
    "world"
  ],
  "channel": {
    "accountcode": "",
    "caller": {
      "name": "blink",
      "number": "blink"
    },
    "connected": {
      "name": "",
      "number": ""
    },
    "creationtime": "2013-10-15T15:54:12.625-0500",
    "dialplan": {
      "context": "default",
      "exten": "7000",
      "priority": 2
    },
    "id": "1381870452.0",
    "name": "SIP/blink-00000000",
    "state": "Ring"
  },
  "timestamp": "2013-10-15T15:54:12.626-0500",
  "type": "StasisStart"
}
```

The `StasisStart` event shows that a channel has entered into the `Stasis` dialplan application. This application is `hello`, and the arguments passed to the application are `["world"]`. You also receive a `channel` object with detailed information about the channel. The channel's `id` field is how the channel is identified via the REST API.

Manipulating the channel



Currently, Asterisk only parses query parameters, even with POST requests. See [ASTERISK-22685](#) for more details.

There are lots of things you can do with channels.

```
$ CHAN=1381870452.0
$ curl -v -u hey:peekaboo -X POST "http://localhost:8088/ari/channels/$CHAN/answer"
# <snip/>
< HTTP/1.1 204 No Content
< Server: Asterisk/SVN-dlee-playback-events-hack-URL:-r400821M
< Date: Tue, 15 Oct 2013 21:10:33 GMT
< Connection: close
< Cache-Control: no-cache, no-store
< Content-Length: 0
<
* Closing connection #0
```

The `/answer` operation simply returns `204 No Content` to indicate success. You will also see a `ChannelStateChange` message on the WebSocket, indicating that the channel is now `"state": "Up"`. If you press DTMF keys on the phone, you will also see `ChannelDtmfReceived` events on the WebSocket.

Asynchronous operations

Most operations you perform via ARI are asynchronous. This means that the API call returns immediately, and work is queued up to happen in the background. For some operations (like `/answer`) this is fairly transparent. For others (like `/play` and `/record`), this is apparent in the API.

Let's look at how playback works.

```
$ curl -v -u hey:peekaboo -X POST
"http://localhost:8088/ari/channels/$CHAN/play?media=sound:hello-world" | jsonpp
# <snip/>
< HTTP/1.1 201 Created
< Server: Asterisk/SVN-dlee-playback-events-hack-URL:-r400821M
< Date: Tue, 15 Oct 2013 21:14:56 GMT
< Connection: close
< Cache-Control: no-cache, no-store
< Content-Length: 146
< Location: /playback/9315bf40-ac65-4cbe-83ae-b4e4355f585e
< Content-type: application/json
<
{ [data not shown]
100 146 100 146 0 0 34515 0 --:--:-- --:--:-- --:--:-- 142k
* Closing connection #0
{
  "id": "9315bf40-ac65-4cbe-83ae-b4e4355f585e",
  "media_uri": "sound:hello-world",
  "target_uri": "channel:1381871629.2",
  "language": "en",
  "state": "queued"
}
```

POST'ing to the `/play` resource returns a `201 Created`, because it creates a `/playback` resource. The URI is given in the `Location:` header, and the `id` is given in the response.

We're now done with the channel. We can hangup with an HTTP DELETE

```
$ curl -v -u hey:peekaboo -X DELETE "http://localhost:8088/ari/channels/$CHAN"
# <snip/>
< HTTP/1.1 204 No Content
< Server: Asterisk/SVN-dlee-playback-events-hack-URL:-r400821M
< Date: Tue, 15 Oct 2013 21:23:12 GMT
< Connection: close
< Cache-Control: no-cache, no-store
< Content-Length: 0
```

Once again, since there's nothing more to say than 'success', the response is 204 No Content.

Bridging

One of the more powerful features in ARI is the ability to create your own bridges, and move channels into and out of those bridges. Let's see some of that now. First, we'll create a holding bridge. This is a bridge which plays music on hold to all participants, and they cannot hear one another.

```
$ curl -v -u hey:peekaboo -X POST "http://localhost:8088/ari/bridges?type=holding"
{
  "channels": [],
  "id": "5b55d1f0-2edf-4b94-a07c-d841e25bba1e",
  "technology": "holding_bridge",
  "bridge_class": "base",
  "bridge_type": "holding"
}
$ BRIDGE="5b55d1f0-2edf-4b94-a07c-d841e25bba1e"
```

There you go, a brand new bridge with no channels in it. If you issue a GET on `/bridges`, or `/bridges/$BRIDGE`, you can see the details of the bridge, including the ids of all the channels currently in the bridge.

Now dial the 7000 extension a couple of times to get a few channels.

Received events

```

{
  "application": "hello",
  "args": [
    "world"
  ],
  "channel": {
    "accountcode": "",
    "caller": {
      "name": "blink",
      "number": "blink"
    },
    "connected": {
      "name": "",
      "number": ""
    },
    "creationtime": "2013-10-15T16:42:28.808-0500",
    "dialplan": {
      "context": "default",
      "exten": "7000",
      "priority": 2
    },
    "id": "1381873348.4",
    "name": "SIP/blink-00000004",
    "state": "Ring"
  },
  "timestamp": "2013-10-15T16:42:28.809-0500",
  "type": "StasisStart"
}
{
  "application": "hello",
  "args": [
    "world"
  ],
  "channel": {
    "accountcode": "",
    "caller": {
      "name": "blink",
      "number": "blink"
    },
    "connected": {
      "name": "",
      "number": ""
    },
    "creationtime": "2013-10-15T16:42:40.572-0500",
    "dialplan": {
      "context": "default",
      "exten": "7000",
      "priority": 2
    },
    "id": "1381873360.5",
    "name": "SIP/blink-00000005",
    "state": "Ring"
  },
  "timestamp": "2013-10-15T16:42:40.573-0500",
  "type": "StasisStart"
}

```

Answer the channels, and add them to the bridge

```
$ CHAN1=1381873348.4
$ CHAN2=1381873360.5
$ curl -u hey:peekaboo -X POST "http://localhost:8088/ari/channels/$CHAN1/answer"
$ curl -u hey:peekaboo -X POST "http://localhost:8088/ari/channels/$CHAN2/answer"
$ curl -v -u hey:peekaboo -X POST
"http://localhost:8088/ari/bridges/$BRIDGE/addChannel?channel=$CHAN1,$CHAN2"
# <snip/>
< HTTP/1.1 204 No Content
```

You should now hear music-on-hold on each channel. Even while in the bridge, you can play media to the channels. Media may be played individually, or to the entire bridge. The music-on-hold will resume when the playback has ended.

```
# Play to the channels individually
$ curl -s -u hey:peekaboo -X POST
"http://localhost:8088/ari/channels/$CHAN1/play?media=sound:hello-world"
$ curl -s -u hey:peekaboo -X POST
"http://localhost:8088/ari/channels/$CHAN2/play?media=sound:hello-world"
# Play to the bridge
$ curl -s -u hey:peekaboo -X POST
"http://localhost:8088/ari/bridges/$BRIDGE/play?media=sound:hello-world"
```

Recommended Practices

Don't access ARI directly from a web page

It's very convenient to use ARI directly from a web page for development, such as using Swagger-UI, or even abusing the [WebSocket echo demo](#) to get at the ARI WebSocket.

But, *please*, do not do this in your production applications. This would be akin to accessing your database directly from a web page. You need to hide Asterisk behind your own application server, where you can handle security, logging, multi-tenancy and other concerns that really don't belong in a communications engine.

Use an abstraction layer

One of the beautiful things about ARI is that it's so easy to just bang out a request. But what's good for development isn't necessarily what's good for production.

Please don't spread lots of direct HTTP calls throughout your application. There are cross-cutting concerns with accessing the API that you'll want to deal with in a central location. Today, the only concern is authentication. But as the API evolves, other concerns (such as versioning) will also be important.

Note that the abstraction layer doesn't (and shouldn't) be complicated. Your client side API can even be something as simple wrapper around GET, POST and DELETE that addresses the cross-cutting concerns. The Asterisk TestSuite has a very simple abstraction library that can be used like this:

```
ari = ARI('localhost', ('username', 'password'))

# Hang up all channels
channels = ari.get('channels')
for channel in channels:
    ari.delete('channels', channel['id'])
```

Higher level abstractions would also be good, but require a more complicated client library. ARI is still very new, but these are [coming along](#).

WebSocket client libraries

- Python

- [websocket-client](#)
- Java
 - [Jetty WebSocket Client API](#)
- Ruby
 - [faye-websocket](#)
- JavaScript
 - [ws](#) (includes a wscat implementation)
- Perl
 - [Net::Async::WebSocket::Client](#)

Create a new resource with ARI

Creating new ARI resources is fairly straightforward.

- [Create the API declaration](#)
- [Add it to `resources.json`](#)
- [Generate the code](#)
- [Implement the API](#)
- [Recommended practices](#)
 - [Use HTTP error codes](#)
 - [Validate your inputs](#)
 - [Don't put business logic in the ARI code](#)

Create the API declaration

In the Asterisk source tree, the Swagger API declarations are stored in `./rest-api/api-docs/`. For this example, we are creating a new resource named "fizzbuzz".

These API declarations are documented using [Swagger](#). Details on documenting the API declarations can be found [on the Swagger wiki](#).

fizzbuzz.json

```
{
  "_copyright": "Copyright (C) 2013, Digium, Inc.",
  "_author": "David M. Lee, II <dlee@digium.com>",
  "_svn_revision": "$Revision$",
  "apiVersion": "0.0.1",
  "swaggerVersion": "1.1",
  "basePath": "http://localhost:8088/stasis",
  "resourcePath": "/api-docs/fizzbuzz.{format}",
  "apis": [
    {
      "path": "/fizzbuzz",
      "description": "The FizzBuzz test. See
http://www.codinghorror.com/blog/2007/02/why-cant-programmers-program.html.",
      "operations": [
        {
          "httpMethod": "GET",
          "summary": "Returns an array of numbers from 1 to 100. But for multiples of three
return \"Fizz\" instead of the number and for the multiples of five return \"Buzz\". For
numbers which are multiples of both three and five return \"FizzBuzz\".",
          "nickname": "fizzbuzz",
          "responseClass": "object",
          "parameters": [
            {
              "name": "max",
              "description": "Set the max number to fizzbuzz up to",
              "paramType": "query",
              "required": false,
              "dataType": "long"
            }
          ]
        }
      ]
    }
  ],
  "models": {
    "FizzBuzz": {
      "id": "FizzBuzz",
      "description": "List of ints, with Fizz and Buzz mixed in",
      "properties": {
        "fizzbuzz": {
          "type": "List[object]"
        }
      }
    }
  }
}
```

Add it to resources.json

The master list of resources served by Asterisk is kept in `rest-api/resources.json`. Simply add your resource to the end of the list.

resources.json.diff

```
Index: rest-api/resources.json
=====
--- rest-api/resources.json (revision 401118)
+++ rest-api/resources.json (working copy)
@@ -41,6 +41,10 @@
     {
       "path": "/api-docs/applications.{format}",
       "description": "Stasis application resources"
+    },
+    {
+      "path": "/api-docs/fizzbuzz.{format}",
+      "description": "FizzBuzz example"
     }
   ]
 }
```

Generate the code

The API declarations are used to generate much of the boilerplate code in Asterisk for routing RESTful API invocations. This code is generated using `make ari-stubs`.



The code generator requires [Pystache](#), which can be installed using `pip install pystache`.

```
$ make ari-stubs
/usr/bin/python rest-api-templates/make_ari_stubs.py \
  rest-api/resources.json .
Writing ./doc/rest-api/Asterisk 12 Fizzbuzz REST API.wiki
Writing ./res/res_ari_fizzbuzz.c
Writing ./res/ari/resource_fizzbuzz.h
Writing ./res/ari/resource_fizzbuzz.c
Writing ./res/ari.make
```

Implement the API

As you can see, a number of files are generated. Most of the files are always regenerated, and not meant to be modified. However `./res/ari/resource_fizzbuzz.c` is simply stub functions to help you get started with your implementation.

The parameters described in your API declaration are parsed into an `args` structure for use in your implementation. The `response` struct is to be filled in with the HTTP response.

resource_fizzbuzz.c

```
void ast_ari_fizzbuzz(struct ast_variable *headers,
    struct ast_fizzbuzz_args *args,
    struct ast_ari_response *response)
{
    RAII_VAR(struct ast_json *, json, NULL, ast_json_unref);
    struct ast_json *fb;
    int i;
    int max = 100;
    if (args->max) {
        max = args->max;
    }
    json = ast_json_pack("{s: []}", "fizzbuzz");
    fb = ast_json_object_get(json, "fizzbuzz");
    /* This is what one would call "business logic", and doesn't belong in
     * the ARI layer. But this is just a silly example.
     */
    for (i = 1; i <= max; ++i) {
        if (i % 15 == 0) {
            ast_json_array_append(fb,
                ast_json_string_create("FizzBuzz"));
        } else if (i % 5 == 0) {
            ast_json_array_append(fb,
                ast_json_string_create("Buzz"));
        } else if (i % 3 == 0) {
            ast_json_array_append(fb,
                ast_json_string_create("Fizz"));
        } else {
            ast_json_array_append(fb, ast_json_integer_create(i));
        }
    }
    ast_ari_response_ok(response, json);
}
```

Recommended practices

Use HTTP error codes

The HTTP error codes do a surprisingly good job describing error conditions you are likely to encounter. Do your best to stay true to the original intention of the error code; it will help keep the API understandable.

The use of extensions can also be useful. For example, we use `422 Unprocessable Entity` to indicate that a request was syntactically correct, but semantically invalid. This helps to keep `400 Bad Request` from being a catch all for all sort of errors.

Validate your inputs

While Swagger can describe input constraints (min, max, required), these are currently not validated in the request routing. Path parameters cannot be `NULL` (because you couldn't route the request if they were), but query parameters could be.

Don't put business logic in the ARI code

The design of Asterisk, including ARI, is to be modular. All of the `res_ari_*.so` modules are supposed to be the logic exposing underlying API's via an HTTP interface. Think of it as a controller in a Model-View-Controller architecture. This could should look up objects, validate inputs, call functions on those object, build the HTTP response.

If you find yourself writing lots of logic in your ARI code, it should probably be extracted down into either a `res_stasis*.so` module, or into Asterisk core.

Asterisk 12 Specifications

The following specifications are provided for Asterisk 12:

- Asterisk Manager Interface (AMI) version 1.4
- Asterisk 12 Call Detail Records (CDR)
- Asterisk 12 Channel Event Logging (CEL)

These specifications are only valid for Asterisk 12 and should not be used as a basis for any other version of Asterisk.

AMI v2 Specification

- Introduction
 - Scope
 - Versioning
- Terminology
- Protocol Overview
- Semantics and Syntax
 - Message Sending and Receiving
 - Message Layout
 - Common Fields
 - Actions
 - Events
 - Channel Interaction/Lifetime
 - Basic Channel Lifetime
 - Channel Variables
 - DTMF
 - Dialplan Execution
 - Dialing and Origination
 - Bridging
 - Two Party Bridging
 - Transfers
 - Local Channel Optimization
 - Masquerades
- Transports
- Security Considerations
 - Class Authorizations
 - Access Control Lists
 - Authorization
- AMI Configuration
 - General Settings
 - Client Settings

Introduction

This Asterisk Manager Interface (AMI) specification describes the relationship between Asterisk and an external entity wishing to communicate with Asterisk over the AMI protocol. It describes:

- An overview of the AMI protocol
- The operations AMI provides external entities wishing to control Asterisk
- Basic formatting of AMI message structures
- Guaranteed operations, configuration control, and other information provided by Asterisk in AMI v2.x

Scope

This specification describes AMI version 2, specifically for Asterisk 12. This specification provides details on the functional, operational and design requirements for AMI version 2. Note that this does not include a comprehensive listing of the AMI configuration file parameters or messages that a system interfacing over AMI in Asterisk 12 will send/receive; however, it does provide a baseline of the supported features and messages provided in AMI version 2. This specification should be used in conjunction with the documented AMI actions and events in Asterisk 12 to encompass the full range of functionality provided by AMI in Asterisk 12.

In addition, this specification provides interface requirements levied on AMI by Stasis, a message bus internal to Asterisk. It conveys sufficient detail to understand how AMI attaches to the Stasis message bus and interacts with other entities on Stasis.

This specification is intended for all parties requiring such information, including software developers, system designers and testers responsible for implementing the interface.

Versioning

Starting in Asterisk 12, AMI now follows semantic versioning. The initial release of Asterisk 12 has an AMI version of 2.0.0.

A full description of semantic versioning can be found at <http://semver.org/>.

Terminology

Term	Definition
Action	A command issued to Asterisk from an external entity via AMI
Client	An external entity communicating with Asterisk via AMI over some transport mechanism
Event	A message sent from Asterisk to an external entity via AMI
Field	A key/value pair that exists in either an action or event
Stasis	The internal framework that AMI is built on top of

Protocol Overview

Asterisk provides a number of interfaces that serve different purposes. Say, for example, we wanted to manipulate a call between Alice and Bob via some external mechanism. Depending on what we wanted to do with the call, we may use one or more interfaces to manipulate the channels that make up the call between Alice and Bob.

Alice calls Bob and...	Interface
... we want to use a local script to execute some logic on Alice's channel	AGI
... we want to execute a script on a remote machine on Bob's channel	FastAGI
... we want to put Alice into an IVR with fine grained media control, where the IVR is written outside of <code>extensions.conf</code>	ExternalIVR
... we want to control Alice and Bob's underlying channel objects at some asynchronous time	AMI (possibly with AsyncAGI)
... we want to write our own Dialling application to control both Alice and Bob	ARI

In general, AMI is used to manage Asterisk and its channels. It does not determine what actions are executed on a particular channel - the dialplan and/or an AGI interface does that - but it does allow a client to control call generation, aspects of call flow, and other internals of Asterisk.

At its heart, AMI is an asynchronous message bus: it spills **events** that contain information about the Asterisk system over some transport. In response, clients may request that Asterisk takes some **action**. These two concepts - actions and events - make up the core of what is AMI. As AMI is asynchronous, as events occur in Asterisk they are immediately sent to the clients. This means that actions issued by entities happen without any synchronization with the events being received, even if those events occur in response to an action. It is the responsibility of entities to associate event responses back to actions.

Clients wishing to use AMI act as clients and connect to Asterisk's AMI server over a supported transport mechanism. Authentication may or may not be enabled, depending on the configuration. Once connected, events can be automatically spilled to the connected clients, or limited in a variety of fashions. A connected client can send an action to the AMI server at any time. Depending on the allowed authorizations, the action may be allowed or disallowed.

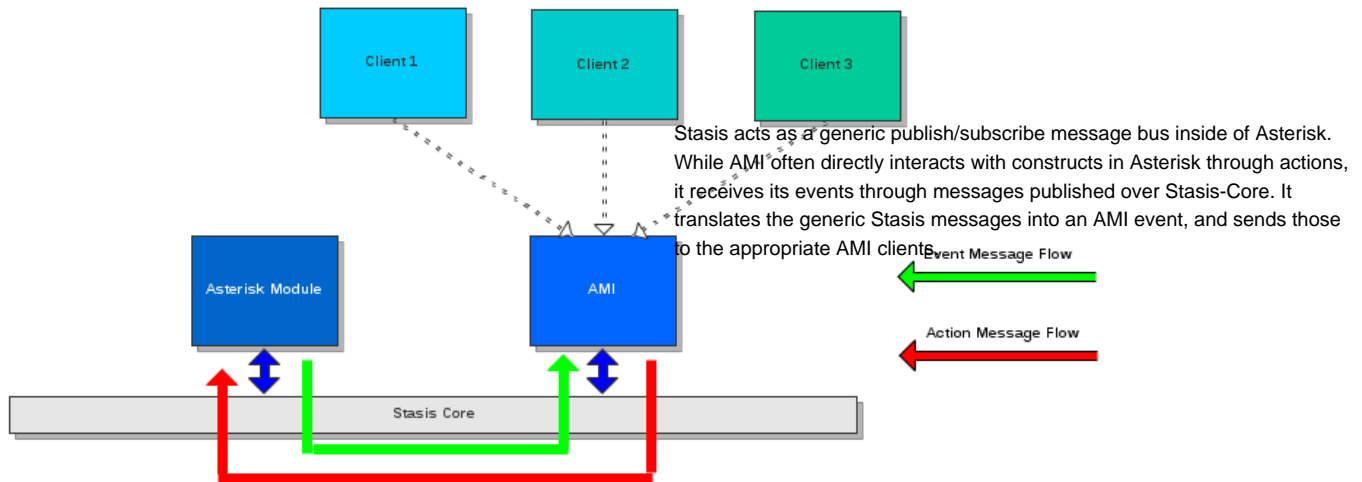
More information on the various ways a client can be configured can be seen in [AMI Configuration](#).



Sometimes, the term **command** may be used instead of the term **action**. With respect to AMI actions, command is synonymous with action, and the two can be treated the same. For the sake of consistency, we've attempted to use the term **action** where possible.

Historically, AMI has existed in Asterisk as its own core component `manager`. AMI events were raised throughout Asterisk encoded in an AMI specific format, and AMI actions were processed and passed to the functions that implemented the logic. In Asterisk 12, AMI has been refactored to sit on top of Stasis, a generic, protocol independent message bus internal to Asterisk. From the perspective of clients wishing to communicate with Asterisk over AMI

very little has changed; internally, the Stasis representation affords a much higher degree of flexibility with how messages move through Asterisk. It also provides a degree of uniformity for information that is propagated to interested parties.



Semantics and Syntax

Message Sending and Receiving

By default, AMI is an asynchronous protocol that sends events immediately to clients when those events are available. Likewise, clients are free to send actions to AMI at any time, which may or may not trigger additional events. The exception to this is when the connection is over HTTP; in that scenario, events are only transmitted as part of the response to an HTTP POST.

Various options for configuration of clients can control which events are sent to a client. Events can be whitelisted/blacklisted explicitly via event filters, or implicitly by class authorizations.

Message Layout

AMI is an ASCII protocol that provides bidirectional communication with clients. An AMI message – action or event – is composed of fields delineated by the '\r\n' characters. Within a message, each field is a key value pair delineated by a ':'. A single space MUST follow the ':' and precede the value. Fields with the same key may be repeated within an AMI message. An action or event is terminated by an additional '\r\n' character.

```
Event: Newchannel
Privilege: call,all
Channel: PJSIP/misspiggy-00000001
Uniqueid: 1368479157.3
ChannelState: 3
ChannelStateDesc: Up
CallerIDNum: 657-5309
CallerIDName: Miss Piggy
ConnectedLineName:
ConnectedLineNum:
AccountCode: Pork
Priority: 1
Exten: 31337
Context: inbound
```

This is syntactically equivalent to the following ASCII string:

```

Event:
Newchannel\r\nPrivilege:
call,all\r\nChannel:
PJSIP/misspiggy-000000
01\r\nUniqueid:
1368479157.3\r\nChannelState:
3\r\nChannelStateDesc:
Up\r\nCallerIDNum:
657-5309\r\nCallerIDName: Miss
Piggy\r\nConnectedLineName:\r\nConnectedLineNum:\r\nAccountCode:
Pork\r\nPriority:\r\nExtension:
31337\r\nContext:
inbound\r\n\r\n

```

Actions are specified in a similar manner. Note that depending on the message, some keys can be repeated.

```

Action: Originate
ActionId: SDY4-12837-123878782
Channel: PJSIP/kermi-00000002
Context: outbound
Exten: s
Priority: 1
CallerID: "Kermit the Frog" <123-4567>
Account: FrogLegs
Variable: MY_VAR=frogs
Variable: HIDE_FROM_CHEF=true

```

In addition, no ordering is implied on message specific keys. Hence, the following two messages are semantically the same.

```

Action: Originate
ActionId: SDY4-12837-123878782
Channel: PJSIP/kermi-00000002
Context: outbound
Exten: s
Priority: 1
CallerID: "Kermit the Frog" <123-4567>
Account: FrogLegs
Variable: MY_VAR=frogs
Variable: HIDE_FROM_CHEF=true

```

```

ActionId: SDY4-12837-123878782
Action: Originate
Variable: HIDE_FROM_CHEF=true
Variable: MY_VAR=frogs
Channel: PJSIP/kermi-00000002
Account: FrogLegs
Context: outbound
Exten: s
CallerID: "Kermit the Frog" <123-4567>
Priority: 1

```

This is also true for events, although by convention, the `Event` key is the first key in the event. If an action or event contains duplicate keys, such as `Variable`, the order in which Asterisk processes said keys is the order in which they occur within the action or event.

Keys are case insensitive. Hence, the following keys are equivalent:

```
Action: Originate
```

```
ACTION: Originate
```

```
action: Originate
```

The case sensitivity for values is left up to the context in which they are interpreted.

Common Fields

Actions

General Fields

This section lists fields that apply generally to all actions.

Action

Action specifies the action to execute within Asterisk. Each value corresponds to a unique action to execute within Asterisk. The value of the **Action** field determines the allowed fields within the rest of the message. By convention, the first field in any action is the **Action** field.

ActionId

ActionId is a universal unique identifier that can optionally be provided with an action. If provided in an action, events that are related to that action will contain the same **ActionId** value, allowing a client to associate actions with events that were caused by that action.

It is recommended that clients always provide an **ActionId** for each action they submit.



It is up to the client to ensure that the **ActionId** provided with an **Action** is unique.

Channels

This section lists fields that apply generally to all actions that interact upon an Asterisk channel. Note that an Action that interacts with a channel *must* supply the **Channel** field.



Upgrading

In the past, AMI clients would have to contend with channel rename events. As Asterisk will now no longer change the name of a channel during its lifetime, this is no longer necessary.

Channel

The Asterisk channel name. A channel name is provided by AMI to clients during a **Newchannel** event. A channel name can be viewed as the handle to a channel.

Uniqueid

A universal unique identifier for the channel. In systems with multiple Asterisk instances, this field can be used to construct a globally unique identifier for a channel, as a channel name may occur multiple times across Asterisk instances.

Events

General Fields

This section lists fields that apply generally to all events.

Event

The unique name of the event being raised. The value of the **Event** field determines the rest of the contents of the message. By convention, the **Event** field is the first field in an AMI message.

ActionId

If present, the Action's corresponding [ActionId](#) that caused this event to be created. If an Action contained an **ActionId**, any event relating the success or failure of that action **MUST** contain an **ActionId** field with the same value.

Privilege

The class authorizations associated with this particular event. The class authorizations for a particular event are in a comma-delineated list. For more information, see [class authorizations](#).

Event responses to an Action only occur if the Action was executed, which means the user had the appropriate class authorization. Therefore they will not

have a Privilege field.

Channels

This section lists fields that apply generally to all events that occur due to interactions upon an Asterisk channel.

Events that relate multiple channels will prefix these fields with an event specific role specifier. For example, a **DialBegin** or a **DialEnd** event will prefix the outbound channel's fields with **Dest**. So, the **Channel** field is the **DestChannel** field; the **Uniqueid** field is the **DestUniqueid** field, etc.

Channel

The current Asterisk channel name. This corresponds to the [Channel](#) field in actions.

Uniqueid

A universal unique identifier for the channel. This corresponds to the [Uniqueid](#) field in actions.

ChannelState

The current state of the channel, represented as an integer value. The valid values are:

Value	State	Description
0	Down	Channel is down and available.
1	Rsrvd	Channel is down, but reserved.
2	OffHook	Channel is off hook.
3	Dialing	The channel is in the midst of a dialing operation.
4	Ring	The channel is ringing.
5	Ringing	The remote endpoint is ringing. Note that for many channel technologies, this is the same as Ring.
6	Up	A communication path is established between the endpoint and Asterisk.
7	Busy	A busy indication has occurred on the channel.
8	Dialing Offhook	Digits (or equivalent) have been dialed while offhook.
9	Pre-ring	The channel technology has detected an incoming call and is waiting for a ringing indication.
10	Unknown	The channel is an unknown state.



Depending on the underlying channel technology, not all states will be used. Channels typically begin in either the Down or Up states.

ChannelStateDesc

The text description of the channel state. This will be one of the State descriptions in the table in [ChannelState](#).

CallerIDNum

The current caller ID number. If the caller ID number is not known, the string "<unknown>" is returned instead.

CallerIDName

The current caller ID name. If the caller ID name is not known, the string "<unknown>" is returned instead.

ConnectedLineNum

The current connected line number. If the connected line number is not known, the string "<unknown>" is returned instead.

ConnectedLineName

The current connected line name. If the connected line name is not known, the string "<unknown>" is returned instead.

AccountCode

The channel's accountcode.

Context

The current context in the dialplan that the channel is executing in.

Exten

The current extension in the dialplan that the channel is executing in.

Priority

The current priority of the current context, extension in the dialplan that the channel is executing in.

ChanVariable

Channel variables specific to a channel can be conveyed in each AMI event related to that channel. When this occurs, each variable is referenced in a **ChanVariable** field. The value of a **ChanVariable** field will always be of the form `key=value`, where `key` is the name of the channel variable and `value` is its value.

Bridges

BridgeUniqueid

A unique identifier for the bridge, which provides a handle to actions that manipulate bridges.

BridgeType

The type of the bridge. Bridge types determine how a participant in a bridge can behave. For example, a 'base' bridge is a bridge that has few inherent properties or features associated with it, while a 'parking' bridge is one used for a parking application. Specific modules within Asterisk will determine the type of bridge that is created.

Note that this is not the same as how media within a bridge is mixed. How media is mixed between participants in a bridge is determined by the **BridgeTechnology**.

BridgeTechnology

How the media can be mixed within a bridge. Specific modules in Asterisk provide different bridge technologies that can be used by Asterisk to alter how media passes between the participants. For a given bridge, the **BridgeTechnology*** can also change as the number and type of participants change. The most common bridge technologies are:

- `holding_bridge` – normal participants joining the bridge may receive audio, but audio sent from a normal participant is dropped. Special participants, known as announcers, may broadcast their audio to all normal participants.
- `native_dahdi` – a native bridge between DAHDI channels. Media is passed directly between all participants.
- `native_rtp` – a native bridge between channels that use RTP for media. Media is passed directly between all participants.
- `simple_bridge` – a two-party bridge between any two channels. Media is passed through the Asterisk core between the two participants.
- `softmix` – a multi-party bridge between one or more participants. All media from all participants is mixed together and sent to all participants.

BridgeCreator

Some bridges are created as the result of a particular application or action. If so, the bridge will specify who created it. If the bridge was not created as a result of any particular application or action, the field will have the value `<unknown>`.

BridgeName

Some bridges are created with a names as a result of their application. If so, the bridge will specify the name given to it. If the bridge was created without a name, the field will have the value `<unknown>`.

BridgeNumChannels

The number of channels currently in the bridge.

Action Responses

When an Action is submitted to AMI, the success or failure of the action is communicated in subsequent events.

Response

Contains whether or not the action succeeded or failed. Valid values are "Success" or "Error". Events that are in response to an action **MUST** include this field.

EventList

Some actions will cause a chain of events to be created. Events that are a response to an action that causes such a sequence will contain the EventList field with a value of "start". When all generated events have been sent, a final event will be sent containing the EventList field with the value "complete".

If, for some reason, an error occurs and the events cannot be sent, an event will be sent with an EventList field that contains the value "cancelled".

Note that the events that mark the completion or cancellation of an event list are not technically action responses, and have their own specific event types.

Message

An optional text message that provides additional contextual information regarding the success or failure of the action.

Actions

The supported actions for Asterisk 12 are listed here:

[Asterisk 12 AMI Actions](#)

While new AMI Actions may be added over the lifetime of Asterisk 12, existing AMI Actions will **not** be removed.

Optional fields may be added to an existing AMI action with altering the AMI version number, but required fields will **not** be added or removed.

Events

The supported events for Asterisk 12 are listed here:

[Asterisk 12 AMI Events](#)

While new AMI Events may be added over the lifetime of Asterisk 12, existing AMI Events will **not** be removed.

Fields may be added to an existing AMI event without altering the AMI version number, but existing fields will **not** be removed.

Channel Interaction/Lifetime

While channels are independent of AMI, they have a large implication on the events sent out over AMI. Many of the events in AMI correspond to changes in channel state. While AMI is an asynchronous protocol, there is some ordering with respect to the events that are relayed for a particular channel. This section provides the basic event relationships that are guaranteed through AMI.

Basic Channel Lifetime

All channels begin with a **Newchannel** event. A **Newchannel** will always contain the following fields:

- The current [Channel](#) name that acts as a handle to the channel for that channel's lifetime for a single Asterisk system.
- The [Uniqueid](#) for the channel, that allows systems to have a globally unique identifier for the channel.

Changes in the state of the channel, i.e., the [ChannelState](#) field, are conveyed via **Newstate** events.

Notification of a Channel being disposed of occurs via a **Hangup** event. A **Hangup** signals the termination of the channel associated with the Uniqueid. After the **Hangup** event, no further events will be raised in relation to the channel with that Uniqueid, and the communication between the endpoint and

Asterisk via that channel is terminated.



The examples in this specification do not show all of the fields in every event. For a full listing of all of the fields, see the documentation for the specific event in [Asterisk 12 AMI Events](#).

Example

```
Event: Newchannel
Privilege: dialplan,all
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479157.1
ChannelState: 0
ChannelStateDesc: Down
...
```

- Kermit the Frog's SIP channel is created. This is the first event for `PJSIP/kermit-00000001` and indicates a path of communication being opened up between Asterisk and Kermit's SIP device.

```
Event: Newstate
Privilege: dialplan,all
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479157.1
ChannelState: 6
ChannelStateDesc: Up
...
```

- Kermit the Frog's PJSIP channel's state changes from Down to Up

```
Event: Hangup
Privilege: dialplan,all
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479157.1
ChannelState: 6
ChannelStateDesc: Up
Cause: 16
Cause-txt: Normal Clearing
...
```

- Kermit the Frog's PJSIP channel is hung up. At this point, no further events for `PJSIP/kermit-00000001` will be sent.

Channel Variables

For each channel variable that is changed, a **VarSet** event is sent to the client. The **VarSet** event contains the new value of the appropriate channel variable. Note that channel variables can also be conveyed in [ChanVariable](#) fields.

DTMF

DTMF is indicated via a **DTMFBegin**/**DTMFEnd** events. A **DTMFEnd** event MUST convey the duration of the DTMF tone in milliseconds.



Behavior Change

The combination of **DTMFBegin**/**DTMFEnd** events replaces the removed **DTMF** event.

Dialplan Execution

As a channel executes operations in the dialplan, those operations are conveyed via a **NewExten** event. Each transition to a new combination of context, extension, and priority will trigger a **NewExten** event.

Example

```
Event: Newexten
Privilege: dialplan,all
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479157.1
Context: default
Extension: h
Priority: 1
Application: NoOp
AppData: Ah Snap
...
```

Kermit the Frog's PJSIP channel has been hung up, and he's been tossed rudely into the *h* extension. This event informs the clients that Kermit's channel is in context *default*, extension *h*, priority *1*, and is about to execute the *NoOp* application with application data *"Ah Snap"*.

Note that even though Kermit has been hungup, we will not receive the **Hangup** event until Kermit's PJSIP channel is done executing dialplan.

Dialing and Origination

Dial operations always result in two events: a **DialBegin** event that signals the beginning of the dial to a particular destination, and a **DialEnd** event that signals the end of the dialing. In parallel dialing situations, **DialBegin/DialEnd** events MUST be sent for each channel dialed. For each **DialBegin** event sent, there MUST be a corresponding **DialEnd** event.

In dialing situations with a caller and a called party, the **DialBegin** and **DialEnd** events convey information about both channels. The calling channel uses the standard channel field names, while the called party's field names are prefixed with "Dest". In dialing situations where there is no caller, such as when Asterisk originates an outbound call via a call file, only the called channel is represented in the events. The channel field names are still prefixed with "Dest" in this case; the standard channel field names are **not** present in the event in this case.

A **DialEnd** occurs whenever Asterisk knows the final state of the channel that it was attempting to establish. The status is communicated in the DialStatus field.



Behavior Change

The **DialBegin/DialEnd** events replace the **Dial** event. Note that the **Dial** event signaling the end of dialing would not normally be sent until after bridging was complete; this operation will now occur when the dial operation has determined the status of a particular called channel.

Simple Successful Dial

```
Event: Newchannel
Channel:
PJSIP/animal-00000
002
Uniqueid:
asterisk-136847916
0.5
...
Event: DialBegin
Channel:
PJSIP/kermit-00000
001
Uniqueid:
asterisk-136847915
5.1
DestChannel:
PJSIP/animal-00000
002
DestUniqueid:
asterisk-136847916
0.5
...
Event: DialEnd
Channel:
PJSIP/kermit-00000
001
Uniqueid:
asterisk-136847915
5.1
DestChannel:
PJSIP/animal-00000
002
DestUniqueid:
asterisk-136847916
0.5
DialStatus: ANSWER
...
```

In this example, Kermit decides to dial Animal. A new channel between Asterisk and Animal's SIP device is created and conveyed via a **Newchannel** event, and then a dial operation is begun. Note that in the **DialBegin** event, Kermit's SIP device is the caller as he initiated the dial operation, while Animal's SIP device is the destination. As such, the fields referring to Animal's PJSIP channel are prefixed with "Dest".

When Animal eats his handset (causing the device to think he merely took it off the hook), the SIP device answers and the dial operation completes. This is indicated by a **DialEnd** event. At this point, the channel is ready for something - it can execute in the dialplan, or be immediately bridged with the calling channel.

Simple Failed Dial

```
Event: Newchannel
Channel: PJSIP/animal-00000003
Uniqueid: asterisk-1368479199.1
...
Event: DialBegin
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479150.1
DestChannel: PJSIP/animal-00000003
DestUniqueid: asterisk-1368479199.1
...
Event: DialEnd
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479150.1
DestChannel: PJSIP/animal-00000003
DestUniqueid: asterisk-1368479199.1
DialStatus: TIMEDOUT
...
Event: Hangup
Channel: PJSIP/animal-00000003
Uniqueid: asterisk-1368479199.1
...
```

In this example, Kermit decides to dial Animal again. Unfortunately, Animal ate his handset, so Asterisk's attempts to reach him inevitably time out. When it does, a **DialEnd** event indicates the failure condition in the DialStatus field.

Parallel Dial

```
Event: Newchannel
Channel: PJSIP/animal-00000003
Uniqueid: asterisk-1368479150.3
...
Event: Newchannel
Channel: PJSIP/drteeth-00000004
Uniqueid: asterisk-1368479150.4
...
Event: DialBegin
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479150.0
DestChannel: PJSIP/animal-00000003
DestUniqueid: asterisk-1368479150.3
...
Event: DialBegin
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479150.0
DestChannel: PJSIP/drteeth-00000004
DestUniqueid: asterisk-1368479150.4
...
Event: DialEnd
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479150.0
DestChannel: PJSIP/drteeth-00000004
DestUniqueid: asterisk-1368479150.4
DialStatus: ANSWER
...
Event: DialEnd
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479150.0
DestChannel: PJSIP/animal-00000003
DestUniqueid: asterisk-1368479150.3
DialStatus: CANCEL
...
Event: Hangup
Channel: PJSIP/animal-00000003
Uniqueid: asterisk-1368479150.3
...
```

In this example, Kermit decides to dial Animal and Dr. Teeth. Dr. Teeth immediately answers, and so Asterisk cancels the dial to Animal. Asterisk can now do something with Dr. Teeth's channel (such as bridge him with his dentist) - however, Animal's channel is destroyed, as his device never answered.

Bridging

A bridge contains 0 or more channels. When a channel is in a bridge, it has the potential to communicate with other channels within the bridge. Before channels enter a bridge, a **BridgeCreate** event is sent, indicating that a bridge has been created. When a bridge is destroyed, a **BridgeDestroy** event is sent. All channels within a bridge **MUST** leave a bridge prior to the **BridgeDestroy** event being sent.

When a channel enters a bridge, a **BridgeEnter** event is raised. When a channel is put into a bridge, it is implied that the channel can pass media between other channels in the bridge. This is not guaranteed, as other properties on the channel or bridge may restrict media flow. For example, bridges with a BridgeTechnology type of *holding_bridge* implicitly restrict the media flow between channels. Likewise, media may be restricted in multi-party conference bridges based on user role permissions, such as when a conference leader mutes all participants in a conference. The **BridgeEnter** event does indicate, however, that a potential relationship between channels in a bridge exists.

When a channel leaves a bridge, a corresponding **BridgeLeave** event is raised. A **BridgeLeave** event MUST mean that the channel that left the bridge can no longer pass media to other channels still in the bridge. This does not necessarily mean that the channel is being hung up; rather, that it is no longer in a communication path with some other set of channels.

In all cases, if a channel has a **BridgeEnter** event, it MUST have a corresponding **BridgeLeave** event. If a channel is hung up and it is in a bridge, a **BridgeLeave** event MUST precede the **Hangup** event.

If a transfer operation is performed, a transfer event of some type MUST be raised for the channels involved in the transfer when the success or failure of the transfer is determined. Similarly, if a channel enters a parking lot, a **ParkedCall** event MUST be raised for the channel prior to it entering the bridge that represents the parking lot.

If a property of a bridge is changed, such as the BridgeTechnology changing from a simple two-party bridge to a multi-party bridge, then the **BridgeUpdate** event is sent with the updated parameters.

Two Party Bridging

Parties are bridged by virtue of them entering a bridge, as indicated by a **BridgeEnter**. When parties are no longer talking, a **BridgeLeave** event is sent for each channel that leaves the bridge.

Example - Two Party Bridge

```
Event: BridgeCreate
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeUniqueid: 1234
BridgeNumChannels: 0
...
Event: BridgeEnter
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeUniqueid: 1234
BridgeNumChannels: 1
Channel: PJSIP/kermi-00000001
Uniqueid: asterisk-1368479150.0
...
```

Kermit the Frog's PJSIP channel enters into Bridge 1234. As a result, the bridge is first created (denoted by the **BridgeCreate** event), and then Kermit's channel enters the bridge (the **BridgeEnter** event)

```
Event: BridgeEnter
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeUniqueid: 1234
BridgeNumChannels: 2
Channel: PJSIP/gonzo-00000002
Uniqueid: asterisk-1368479150.1
...
```

Gonzo the Great enters the bridge and talks with Kermit. Note that the bridge Gonzo entered is Bridge 1234; by virtue of this being the same bridge Kermit entered, we know that the two can talk.

```
Event: BridgeLeave
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeUniqueid: 1234
BridgeNumChannels: 1
Channel: PJSIP/kermi-00000001
Uniqueid: asterisk-1368479150.0
...
Event: Hangup
Channel: PJSIP/kermi-00000001
Uniqueid: asterisk-1368479150.0
...
```

Kermit realizes that he has to leave to avoid Miss Piggy, so he hangs up on Gonzo. We are first alerted that Kermit has left the bridge; quickly thereafter, we receive the **Hangup** event indicating that Kermit's channel is dead.

```
Event: BridgeLeave
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeUniqueid: 1234
BridgeNumChannels: 0
Channel: PJSIP/gonzo-00000002
Uniqueid: asterisk-1368479150.1
...
Event: Hangup
Channel: PJSIP/gonzo-00000002
Uniqueid: asterisk-1368479150.1
...
Event: BridgeDestroy
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeUniqueid: 1234
BridgeNumChannels: 0
...
```

Asterisk is configured to not let Gonzo continue on in the dialplan once his bridge is broken. As such, Gonzo is forcibly ejected from the bridge, and is hung up on after. Because no channels are left in the bridge, the bridge is destroyed.



In this scenario, it was perfectly acceptable for either Kermit or Gonzo's channels to continue after the bridge was broken. Since this represents the most basic two-party call scenario, once one party decided to hang up, the other party was also hung up on.

Transfers

Transfer information is conveyed with either a **BlindTransfer** or **AttendedTransfer** event, which indicates information about the transfer that took place. **BridgeLeave/BridgeEnter** events are used to indicate which channels are talking in which bridges at different stages during the transfer.



Transfers do a Lot

Depending on the type of transfer and the actions taken, channels will move in and out of a lot of bridges. The purpose of the two transfer events is to convey to the AMI client the overall completed status of the transfer after the users have completed their actions. With Blind Transfers, this typically happens very quickly: Asterisk simply has to determine that the destination of the transfer is a valid extension in the dialplan.

Attended transfers, on the other hand, can involve a lot more steps. Parties can consult, toggle back and forth between consultations, and merge bridges together. Channels can be transferred to a dialplan application directly, and not to another party! As such, Asterisk will send the **Attende**

dTransfer event when it knows whether or not the Attended Transfer has completed successfully, and will attempt to convey as much information as possible about the final status of the transfer.

For more information on these events, see [BlindTransfer](#) and [AttendedTransfer](#).

Example - Blind Transfer

```
Event: BridgeCreate
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeUniqueid: 1234
...
Event: BridgeEnter
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeUniqueid: 1234
Channel: PJSIP/kermi-00000001
Uniqueid: asterisk-1368479150.0
...
```

Kermit the Frog's PJSIP channel enters into Bridge 1234

```
Event: BridgeEnter
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeUniqueid: 1234
Channel: PJSIP/fozzie-00000002
Uniqueid: asterisk-1368479150.2
...
```

Fozzie Bear's PJSIP channel enters into Bridge 1234. At this point, Fozzie and Kermit can talk to each other.

```
Event: DTMFBegin
Channel: PJSIP/fozzie-00000002
Uniqueid: asterisk-1368479150.2
Digit: #
Direction: Received
...
Event: DTMFEnd
Channel: PJSIP/fozzie-00000002
Uniqueid: asterisk-1368479150.2
Digit: #
DurationMS: 150
Direction: Received
...
Event: DTMFBegin
Channel: PJSIP/fozzie-00000002
Uniqueid: asterisk-1368479150.2
Digit: 1
Direction: Received
...
Event: DTMFEnd
Channel: PJSIP/fozzie-00000002
Uniqueid: asterisk-1368479150.2
Digit: 1
DurationMS: 150
Direction: Received
...
Event: Hold
Channel: PJSIP/fozzie-00000002
Uniqueid: asterisk-1368479150.2
...
Event: MusicOnHoldStart
Channel: PJSIP/kermi-00000001
Uniqueid: asterisk-1368479150.0
...
Event: Unhold
Channel: PJSIP/fozzie-00000002
Uniqueid: asterisk-1368479150.2
...
Event: BlindTransfer
Result: Success
TransfererChannel: PJSIP/fozzie-00000002
TransfererUniqueid: asterisk-1368479150.2
BridgeUniqueid: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
Context: default
Extension: 2000
```

Fozzie decides he's tired of telling Kermit jokes and blind transfers him off to Miss Piggy via the dialplan extension 2000. The following actions take place:

- Fozzie hits #1 on his phone to initiate a blind transfer (shown by the two pairs of **DTMFBegin/DTMFEnd** event s).
- A **Hold** event is processed for Fozzie, indicating that he's attempting to put the participants of the bridge he is in with on hold.
- A **StartMusicOnHold** event occurs for Kermit, as he is now entertained with music while Fozzie dials Miss Piggy's extension.
- Fozzie would then dial Miss Piggy's extension. Note that we didn't bother showing the DTMF events for this. When Asterisk determines that 2000 is a valid extension, a **BlindTransfer** event is raised indicating that a successful transfer is occurring.



When a transfer occurs, we often think of a transferer channel transferring some target channel. This is actually not the case: a transferer transfers the bridge they are currently in to some other location. This is why the **BlindTransfer** (and **AttendedTransfer**) events show the bridge that is being transferred, and not a destination channel. It is possible to transfer multi-party bridges to new extensions in this manner.



At this point, the order of the events that affect Fozzie versus Kermit are not defined. Fozzie and Kermit are about to be no longer bridged together, and their respective events may arrive in any order.

```

Event: MusicOnHoldStart
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479150.0
...
Event: BridgeLeave
BridgeUniqueid: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479150.0
...

```

Kermit leaves the bridge with Fozzie. Asterisk politely turns off the hold music to him before it ejects him from the bridge.

```

Event: BridgeLeave
BridgeUniqueid: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
Channel: PJSIP/fozzie-00000002
Uniqueid: asterisk-1368479150.2
...
Event: BridgeDestroy
BridgeUniqueid: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
...

```

Because Fozzie isn't talking to anyone anymore, he leaves the bridge as well. At this point Asterisk could hang up Fozzie's channel, or, if configured, he could continue on in the dialplan (say, perhaps, to talk to his rubber chicken).

```

Event: NewExten
Channel: PJSIP/kermit-00000001
Uniqueid: asterisk-1368479150.0
Context: default
Exten: 2000
Priority: 1
...

```

Kermit enters into dialplan extension 2000. Presumably, he'll begin calling Miss Piggy, although given her violent temperament, he'd probably be better off hanging up immediately.

Local Channel Optimization

Local channels have an option wherein they can be optimized away if both halves of a Local channel are in a bridge. This option is set on Local channel creation, and is communicated back to the AMI clients in the **LocalBridge** event in the **LocalOptimization** field. When a Local channel optimization occurs, a **LocalOptimizationBegin** event is sent that indicates the channels involved in the optimization. When the optimization has completed and the parties can now converse without the Local channel, a **LocalOptimizationEnd** event is sent.

Two actions can take place when a Local channel optimizes between two bridges.

1. If, after the Local channel optimization, either bridge contains only a single channel, then a single channel in one of the bridges is moved to the bridge that has the other channel. This is conveyed by a sequence of **BridgeLeave/BridgeEnter** events.
2. If, after the Local channel optimization, the bridges contain multiple parties, the bridges will be merged together. A **BridgeMerge** event is sent when this occurs. Channels will then be merged from one bridge to the other, denoted by a sequence of **BridgeLeave/BridgeEnter** events.



It is not defined which channel is moved first or which bridge wins during a bridge merge. That is an implementation detail left up to Asterisk. Suffice to say, if a Local channel is optimized away, Asterisk attempts to rebridge the channels left over as fast as possible to prevent any loss in audio.

Example - Optimizing Local Channel between two PJSIP Channels

```

Event: Newchannel
Channel: PJSIP/gonzo-00000001
Uniqueid: asterisk-1368479150.0
...

```

Gonzo decides to call Kermit the Frog, and a channel is created between Asterisk and Gonzo's SIP device.

```
Event: Newchannel
Channel: Local/kermit@default-00000001;1
Uniqueid: asterisk-1368479150.1
...
Event: Newchannel
Channel: Local/kermit@default-00000001;2
Uniqueid: asterisk-1368479150.2
...
Event: DialBegin
Channel: PJSIP/gonzo-00000001
Uniqueid: asterisk-1368479150.0
DestChannel: Local/kermit@default-00000001;1
DestUniqueid: asterisk-1368479150.1
...
```

Instead of dialing Kermit directly, Gonzo instead dials a Local channel. Both halves of the Local channel are created, and Gonzo Dials the first half of the Local channel.

```
Event: DialEnd
Channel: PJSIP/gonzo-00000001
Uniqueid: asterisk-1368479150.0
DestChannel: Local/kermit@default-00000001;1
DestUniqueid: asterisk-1368479150.1
DialStatus: ANSWER
...
Event: BridgeCreate
BridgeUniqueid: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 0
...
Event: BridgeEnter
BridgeUniqueid: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 1
Channel: PJSIP/gonzo-00000001
Uniqueid: asterisk-1368479150.0
...
Event: BridgeEnter
BridgeUniqueid: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 2
Channel: Local/kermit@default-00000001;1
Uniqueid: asterisk-1368479150.1
...
Event: LocalBridge
LocalOneChannel: Local/kermit@default-00000001;2
LocalOneUniqueid: asterisk-1368479150.1
LocalTwoChannel: Local/kermit@default-00000001;2
LocalTwoUniqueid: asterisk-1368479150.2
LocalOptimization: Yes
...
```

The dial operation succeeds and Gonzo is bridged with one half of the Local channel that will eventually connect him (it?) to Kermit. The Local channel itself has also decided to go ahead and bridge the two local halves together, since we have one half of the full chain of channels established.

```

Event: DialBegin
Channel: Local/kermit@default-00000001;2
Uniqueid: asterisk-1368479150.2
DestChannel: PJSIP/kermit-00000002
DestUniqueid: asterisk-1368479150.3
...
Event: DialEnd
Channel: Local/kermit@default-00000001;2
Uniqueid: asterisk-1368479150.2
DestChannel: PJSIP/kermit-00000002
DestUniqueid: asterisk-1368479150.3
DialStatus: ANSWER
...
Event: BridgeCreate
BridgeUniqueid: 5678
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 0
...
Event: BridgeEnter
BridgeUniqueid: 5678
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 1
Channel: Local/kermit@default-00000001;2
Uniqueid: asterisk-1368479150.2
...
Event: BridgeEnter
BridgeUniqueid: 5678
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 2
Channel: PJSIP/kermit-00000002
Uniqueid: asterisk-1368479150.3
...

```

```

Event: LocalOptimizationBegin
LocalOneChannel: Local/kermit@default-00000001;1
LocalOneUniqueid: asterisk-1368479150.1
LocalTwoChannel: Local/kermit@default-00000001;2
LocalTwoUniqueid: asterisk-1368479150.2
SourceChannel: PJSIP/gonzo-00000001
SourceUniqueid: asterisk-1368479150.0
...
Event: BridgeLeave
BridgeUniqueid: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 1
Channel: PJSIP/gonzo-00000001
Uniqueid: asterisk-1368479150.0
...
Event: BridgeEnter
BridgeUniqueid: 5678
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 3
Channel: PJSIP/gonzo-00000001
Uniqueid: asterisk-1368479150.0
...

```

Gonzo, via the Local channel halves, dials Kermit and he answers. Kermit is now bridged with the second half of the Local channel, Local/kermit@default-00000001;2, while Gonzo is bridged with the first half of the Local channel, Local/kermit@default-00000001;1.



Note that even when Local channels are destined to be optimized away, both the Local channel halves and the other channels involved in the operation first enter their respective bridges.

Asterisk determines that it can optimize away the Local channel. It notifies the AMI client that this is about to begin, which Local channels are involved, and what channel it is going to start moving (the SourceChannel). It starts first by moving Gonzo into Kermit's bridge. From their perspective, nothing has happened - they just now happen to be in the same bridge, instead of having a Local channel pass frames for them.

```

Event: BridgeLeave
BridgeUniqueId: 5678
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 2
Channel: Local/kermit@default-00000001;2
UniqueId: asterisk-1368479150.2
...
Event: LocalOptimizationEnd
LocalOneChannel: Local/kermit@default-00000001;1
LocalOneUniqueId: asterisk-1368479150.1
LocalTwoChannel: Local/dial_bar@default-00000001;2
LocalTwoUniqueId: asterisk-1368479150.2
...
Event: BridgeLeave
BridgeUniqueId: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 0
Channel: Local/kermit@default-00000001;1
UniqueId: asterisk-1368479150.1
...
Event: Hangup
Channel: Local/kermit@default-00000001;1
UniqueId: asterisk-1368479150.1
...
Event: Hangup
Channel: Local/kermit@default-00000001;2
UniqueId: asterisk-1368479150.2
...
Event: BridgeDestroy
BridgeUniqueId: 1234
BridgeType: basic
BridgeTechnology: simple_bridge
BridgeNumChannels: 0
...

```

The Local channel optimization completes by removing the Local channel halves from their respective bridges and hanging them up. Because Bridge 1234 (which used to have Gonzo in it) no longer has anyone in it, it is destroyed.

Masquerades



Masquerades are gone

In the past, masquerades occurred rather frequently - most often in any scenario where a transfer occurred or where a `pbx_thread` needed to be associated with a channel. This has now changed. Masquerades now rarely occur, and are never communicated to AMI clients. From the perspective of AMI clients, nothing changes - you still use your handle to a channel to communicate with it, regardless of the presence (or lack thereof) of a masquerade operation.

This section only exists to explicitly call out the fact that Masquerades are gone.

Transports

AMI supports the following transport mechanisms:

- TCP/TLS
- HTTP/HTTPS

When clients connect over HTTP/HTTPS, AMI events are queued up for retrieval. Events queued up for a client are automatically retrieved and sent in the response to any POST operation. The **WaitEvent** action can be used to wait for and retrieve AMI events.

Security Considerations

AMI supports security at the transport level via TLS using OpenSSL.

For specific security considerations and best practice, please see the [README-SERIOUSLY.bestpractices.txt](#) included with Asterisk.

Class Authorizations





Do not rely on class authorizations for security. While they provide a means to restrict a client's access to sets of functionality, there are often ways of achieving similar functionality through multiple mechanisms. Do **NOT** assume that because a class authorization has not been granted to a client, that they can't find a way around it. In general, view class authorizations as a coarse grained way of providing sets of filters.

Events and actions are automatically classified with particular class authorizations. Clients can be configured to support some set of class authorizations, filtering the actions that they can perform and events that they receive. The supported class authorizations are listed below.

Class Type	Description
system	The item is associated with something that reports on the status of the system or manipulates the system in some fashion
call	The item is associated with calls, i.e., state changes in a call, etc.
log	The item is associated with the logging subsystem
verbose	The item is associated with verbose messages
command	The item is associated with execution of CLI commands through AML
agent	The item is associated with Queue Agent manipulation
user	The item is associated with user defined events
config	The item is associated with manipulating the configuration of Asterisk
dtmf	The item is associated with DTMF manipulation
reporting	The item is associated with querying information about the state of the Asterisk system
cdr	The item is associated with CDR manipulation
dialplan	The item is associated with dialplan execution
originate	The item is associated with originating a channel
agi	The item is associated with AGI execution
cc	The item is associated with call completion
aoc	The item is associated with Advice of Charge
test	The item is associated with some test action
message	The item is associated with out of call messaging
security	The item is associated with a security event in Asterisk
all	The item has all class authorizations associated with it
none	The item has no class authorization associated with it

Access Control Lists

Access Control Lists can be used to filter connections based on address. If an attempt to connect from an unauthorized address is detected, the connection attempt will be rejected.

Authorization

Authorization can be provided via the **Login** action. If a client fails to provide a valid username/password, the connection attempt and any subsequent actions will be rejected. Events will not be sent until the client provides authorized credentials.

The actions that are excluded from successful login are:

- **Login**

- Logoff
- Challenge

AMI Configuration

AMI supports the following configuration options. Note that additional configurations MAY be specified; however, these configuration options are valid for Asterisk 12.

General Settings

Option	Type	Description	Default
enabled	Boolean	Enable AMI	no
webenabled	Boolean	Enable AMI over HTTP/HTTPS	no
port	Integer	The port AMI's TCP server will bind to	5038
bindaddr	IP Address	The address AMI's TCP server will bind to	0.0.0.0
tlsenable	Boolean	Enable TLS over TCP	no
tlsbindaddr	IP Address	The address AMI's TCP/TLS server will bind to	0.0.0.0:5039
tlscertfile	String	The full path to the TLS certificate to use	/tmp/asterisk.pem
tlsprivatekey	String	The full path to the private key. If no path is specified, <i>tlscertfile</i> will be used for the private key.	/tmp/private.pem
tlscipher	String	The string specifying which SSL ciphers to use. Valid SSL ciphers can be found at http://www.openssl.org/docs/apps/ciphers.html#CIPHER_STRINGS	
allowmultiplelogin	Boolean	Allow multiple logins for the same user. If set to no, multiple logins from the same user will be rejected.	Yes
timestampevents	Boolean	Add a Unix epoch *Timestamp* field to all AMI events	No
authlimit	Integer	The number of unauthenticated clients that can be connected at any time	

Client Settings

Note that the name of the client settings context is the username for the client connection.

Option	Type	Description	Default
--------	------	-------------	---------

secret	String	The password that must be provided by the client via the Logi n action.	
deny	ACL	An address/mask to deny in an ACL. This option may be present multiple times.	
permit	ACL	An address/mask to allow in an ACL. This option may be present multiple times.	
acl	String	A Named ACL to apply to the client.	
setvar	String	A channel variable key/value pair (using the nomenclature VARIABLE=value) that will be set on all channels originated from this client	
eventfilter	RegEx	<p>This option may be present multiple times. This options allows clients to whitelist or blacklist events. A filter is assumed to be a whitelist unless preceeded by a '!'. Evaluation of the filters is as follows:</p> <ul style="list-style-type: none"> • 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. 	
read	String	A comma delineated list of the allowed class authorizations applied to events	all
write	String	A comma delineated list of the allowed class authorizations applied to actions	all

The item has all class authorizations associated with it

Asterisk 12 CDR Specification

- Introduction
 - Scope
 - Terminology
 - CDR Overview
 - Semantics and Syntax
 - Basic CDR Lifetime
 - Detailed Semantics
 - CDR Lifetime
 - Creation
 - Dialing Parties
 - Bridging
 - Finalization
 - Dispatch
 - Choosing the Party A channel
 - LinkedID Propagation
 - Fields
 - Standard Fields
 - Dispositions
 - AMA Flags
 - User Defined Fields
 - Scenarios
 - Unanswered "Inbound" Call
 - Unanswered "Outbound" Call
 - Single Party
 - Basic Two Party Calls
 - Basic Call
 - Unanswered Dial
 - Parallel Dial
 - Call Forward
 - Transfers
 - Blind Transfer
 - Attended Transfer to Channel
 - Attended Transfer to Application
 - Blonde Transfer
 - Three Way Call
 - SIP Attended Transfer
 - Local Channels
 - Non-optimizing Local Channels
 - Local channel to an application
 - Local channel between bridges
 - Local Channel Optimization
 - Call Hold
 - Call Park
 - Call Queues
 - Call Queue - Example 1
 - Call Queue - Example 2
 - Conference Call
 - A Complex Example
- Asterisk CDR APIs
 - Applications
 - NoCDR
 - ForkCDR
 - ResetCDR
 - Functions
 - CDR
 - CDR_PROP

- [Example: Setting the called party as Party A of the CDR](#)
- [Configuration](#)
 - [General Settings](#)

Introduction

Call Detail Records, or CDRs, have been around in Asterisk for a long, long time. In fact, portions of it were taken from the Zapata library, as noted in `cdr.c`:

```
*
* \note Includes code and algorithms from the Zapata library.
*
```

CDRs have always attempted to provide the billing information between two parties involved in a call. While their simplicity has been a big advantage in quickly deploying a billing system, it has also been their largest disadvantage when trying to express complex scenarios. As Asterisk evolved and adapted to more complex scenarios, it became difficult to express the full details of a call within a single CDR. Compounding this problem, the way in which bridging was performed in Asterisk made call scenarios involving more than two parties incredibly difficult to track and coalesce into one or more CDRs. Numerous attempts were made to address the various problems in CDRs with varying degrees of success.

In Asterisk 12, changes in the bridging architecture necessitated a substantial upgrade to CDR behavior. The largest changes are:

- Individual components in Asterisk no longer modify CDRs directly. CDRs are produced based on the state of the channels and bridges within Asterisk. As a result, there is a much greater degree of consistency in CDRs throughout Asterisk, regardless of the application channels happen to be executing in.
- The behavior of CDRs between multiple parties is now defined.
- Depending on how channels are dialed and bridged, multiple CDRs will be created for a given call. Post-processing of these records **will** be required to determine the overall statistics of the call.

It is important to note that in [Asterisk 1.8](#), Channel Event Logging (CEL) was introduced as an alternative to CDRs. Unlike CDRs, where a relatively few records are produced for a call, CEL provides a sequence of events regarding the state of channels within Asterisk. CELs provide substantially more information about what is occurring to the channels involved in a call, thus allowing an Asterisk user to construct their own billing system by handling the events as they choose. While CEL will never supplant CDRs, they are an option if CDRs do not provide the billing information you need or in the format you require. While improvements have been made and some complex scenarios defined in this specification, the fact that CEL can provide a more robust billing system is still true as of Asterisk 12.

Asterisk Users moving to Asterisk 12 are highly encouraged to read this specification carefully.

Scope

This CDR specification applies to Asterisk 12. It does not cover CDRs in prior versions of Asterisk.

Note that this does not include a comprehensive analysis as to how CDRs can be produced in all call scenarios in Asterisk. It defines the behavior for common scenarios, but certain scenarios are deliberately left unspecified. The behavior of CDRs in said scenarios is **undefined and is not a bug**.

It is known that the behavior of CDRs will not allow all applications to capture the billing requirements for their systems. If CDRs cannot meet the requirements of your application, [Channel Event Logging](#) (CEL) provides call information at a much finer granularity, allowing complex billing systems to be constructed. Please see the [Asterisk 12 CEL Specification](#) for more information on CEL.

Terminology

Term	Definition
CDR	Call Detail Record. The thing that this document is attempting to describe.
CEL	Channel Event Logging. An alternative way to get billing information from Asterisk, that is significantly more flexible and powerful than CDRs but requires the billing logic to be completely implemented by the user.
Party A	A CDR always involves two parties. One party is always chosen as the 'owner' of the CDR. The CDR reflects the view of the call from that party.
Party B	A CDR always involves two parties. Party B is the target of the call.

Stasis	Stasis is the internal message bus in Asterisk that conveys state to the CDR engine.
--------	--

CDR Overview

A CDR is a record of communication between one or two parties. As such, a single CDR always addresses the communication between two parties: a Party A and a Party B. The CDR reflects the view of the call from the perspective of Party A, while Party B is the party that Party A is communicating with. Each CDR includes the following times:

- Start time - the time at which the CDR was created for Party A
- Answer time - the time at which Party A and Party B could begin communicating
- End time - the time at which Party A and Party B could no longer communicate

From these times, two durations are computed:

- Duration - the End time minus the Start time.
- Billsec - the End time minus the Answer time. (Whether or not you actually bill for this period of time is up to you)

A single CDR only tracks information about a single path of communication between two endpoints. In many scenarios, there will be multiple paths of communication between multiple parties, even in a single "call". Each path of communication results in a new CDR, each representing the communication between two endpoints. All of the CDRs involved are associated by virtue of a special linked identifier field, `linkedid`. The CDRs themselves, however, typically do **not** aggregate the time between records. It is up to billing systems to determine which CDRs should be used for their billing records, and add up the times/durations themselves.

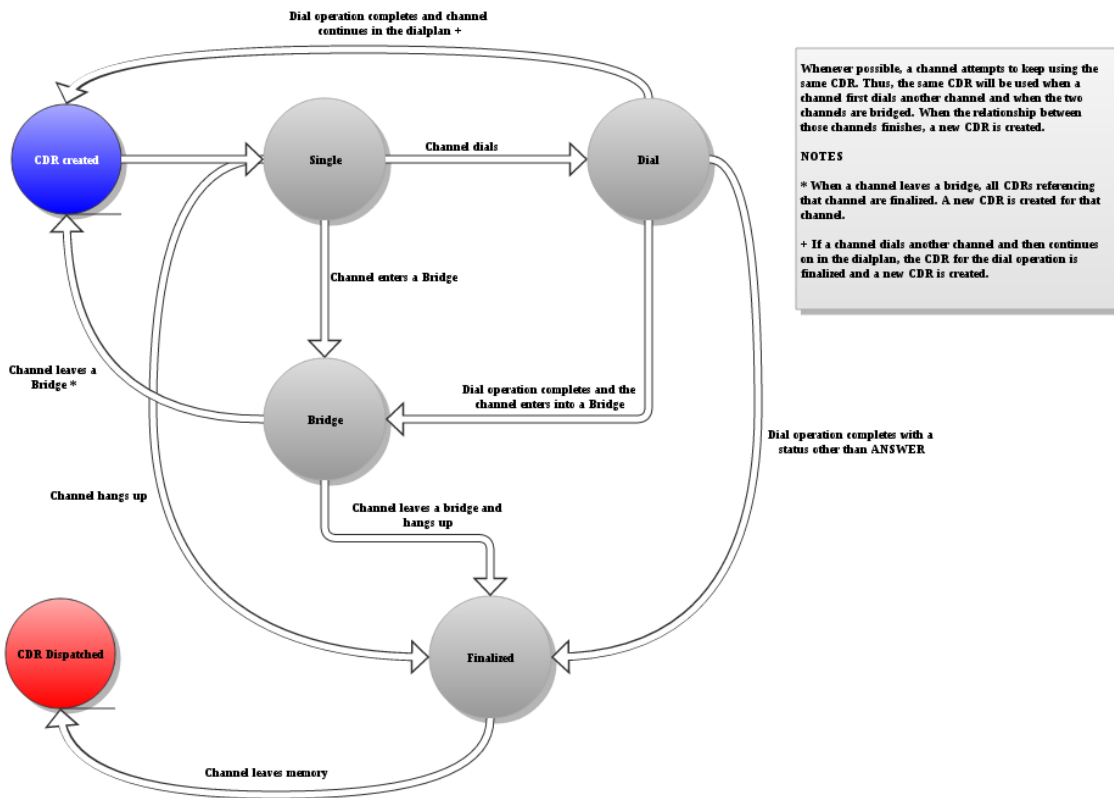
Semantics and Syntax

Basic CDR Lifetime

Fundamentally, a CDR represents a path of communication between a channel and Asterisk or between two channels communicating through Asterisk. Those relationships can be broken down into the following states:

- Single - a channel is executing dialplan in Asterisk or otherwise has no relationship with any other channel. This is the default state of a CDR.
- Dial - a channel is involved in a dial operation - either as a caller or as the callee. If they are the caller, they are automatically the Party A in the CDR. If they are the callee, they are automatically the Party B if there is a caller. If there is no caller - as is the case in channel origination - they are the Party A and there is no Party B.
- Bridge - two channels share a bridge in which they can potentially talk to each other.
- Finalized - the channel and Asterisk can no longer communicate or the relationship between the two channels has been broken. At this point, the last state of the channels involved is locked in the CDR.
- Dispatched - the CDR has been written to the backend storages.

When possible, Asterisk attempts to reuse a CDR. Thus a single CDR can transition through all of the above states. However, once a relationship is established between a Party A and a Party B, the CDR **cannot** be reused. The CDR will be finalized when the relationship between channels is broken and a new CDR created for any channel that is not hung up.



When a channel is created, a CDR is created for that channel. The channel is automatically the Party A of that CDR. The CDR is considered to be in the **Single** state while the channel executes dialplan or is waiting to be associated with another channel.

If a channel dials another channel, the CDR for that channel is transitioned to the **Dial** state. If the dial operation completes successfully and the channels are bridged together, the CDR transitions to the **Bridged** state. If the dial operation fails, the CDR transitions to the **Finalized** state. If the caller continues on in the dialplan, a new CDR is generated for them.

While the channels are bridged together, the CDR remains in the **Bridged** state. Operations that restrict media flow, such as call hold, are not reflected in CDRs. When the bridge is broken - either because one of the parties hangs up or a party is transferred - the CDR transitions to the **Finalized** state and any non-hungup channels have a new CDR created for them.

When a channel hangs up, all CDRs associated with it are implicitly finalized. When all CDRs for a channel are finalized, the CDRs are **Dispatched** to the backends for storage.

Detailed Semantics

CDR Lifetime

Creation

A CDR record is created in any one of the following situations:

- Whenever a channel is created.
- Whenever a channel leaves a bridge and is not hung up.
- When a CDR is forked from a prior record.
- When a channel enters a multi-party bridge.
- When a channel **dials more than one channel**.

When it is created, a CDR inherits the `uniqueid` and `linkedid` from its Party A channel, and a new `sequence` number is generated. When created as a result of a dial operation, the channel acting as the caller is always the Party A.

Dialing Parties

When a channel is known to dial other channels, a CDR is created for each dial attempt. The dial status is recorded for each dial attempt as a [CDR Disposition](#). Note that not all dial attempts may be dispatched depending on the CDR configuration. The caller is always the Party A in the created CDRs.

Bridging

If a channel is bridged with another channel, the following procedure is performed:

- The CDR is marked as having entered a bridge. If there is no other channel in the bridge, the CDR waits for another channel to join.
- For each pairing of channels, the channels are compared to determine which is the Party A channel and which is the Party B channel. (See [Choosing the Party A channel](#)).
 - If the channel entering the bridge is the Party A, the CDR has a Party B, and the channel it is bridged with is the Party B, the CDR continues.
 - If the channel entering the bridge is the Party A, the CDR has a Party B, and the channel is not already the Party B, the current CDR is finalized and a new CDR is created for the relationship between the two parties. Note that the original CDR will be re-activated if the existing Party B enters the bridge.
 - If the channel entering the bridge is the Party A, the CDR has no Party B, then the channel it is bridged with becomes the Party B.
 - If the channel entering the bridge is the Party B, the other channel has a CDR with no Party B, this channel becomes the Party B and the existing CDR is finalized.
 - If the channel entering the bridge is the Party B, the other channel has a CDR with a Party B, and this channel is that CDR's Party B, then the existing CDR is finalized and the other channel's CDR activated.
 - If the channel entering the bridge is the Party B, the other channel has a CDR with a Party B, and this channel is not that CDR's Party B, then the existing CDR is finalized and a new CDR is created for that other channel with this channel as the Party B.
- If a third party joins the bridge with Party A and Party B, the process [Choosing the Party A channel](#) is repeated for each pairing of channels. Thus, in a three-way call there will be three CDR records; in a four-way call there will be six records, etc.



This feels complex, but there's really two rules going on here:

1. Keep using the existing CDR for a channel as long as possible
2. Make CDRs for all pairings of channels in a bridge

Finalization

A CDR is finalized in one of the following scenarios:

- If in a dial, the dial operation completes with a status other than ANSWER
- If in a bridge, either party A or party B leaves the bridge
- Either channel in a CDR hangs up
- The CDR is forked and the forking operation instructs that the CDR should be finalized

When a CDR is finalized, no further modifications can be made to the CDR by the user or Asterisk.

If a Party A channel in a CDR is not hung up but the CDR is finalized - such as when the channel leaves a bridge or its Party B hangs up - a new CDR is made for that channel and the process in [CDR Creation](#) is begun again. Note that if the Party B in a CDR continues on in the dialplan and/or is bridged with a new party, it may become Party A for a new CDR.

If at any point the Party A channel for a CDR is hung up, all CDR records for that Party A are [dispatched](#).

Dispatch

When a CDR is dispatched, all CDRs associated with the channel are committed to permanent storage. The CDRs at this point are removed from memory.

Choosing the Party A channel

Asterisk does not have the concept of "internal" versus "external" devices. As such, what constitutes the Party A channel is highly dependent on a particular system configuration which is outside the control of the CDR system. As such, choosing a Party A uses the following rules:

1. If the channel was dialed (but not originated), the channel is always Party B.
2. If one of the two channels has the `party_a` flag set, then that channel is chosen as the Party A.
3. If neither or both channels have the `party_a` flag, the channel with the oldest creation time is chosen as the Party A.

The `party_A` flag may be set using the [CDR function](#).

LinkedID Propagation

When two channels are bridged, the `linkedid` property for the channels is updated. The channel with the oldest `linkedid` "wins", and the other channel's `linkedid` is replaced. This creates an association between the channels that lasts even if the bridge is broken at a latter time.

Note that dialed channels automatically receive the `linkedid` of the calling channel.

Fields

Standard Fields


Field	Type	Description	Access
accountcode	String (20)	An account code associated with the Party A channel	r/w
src	String (80)	The Caller ID Number	r
dst	String (80)	The destination extension	r
dcontext	String (80)	The destination context	r
clid	String (80)	The Caller ID with text	r
channel	String (80)	The name of the Party A channel	r
dstchannel	String (80)	The name of the Party B channel	r
lastapp	String (80)	The last application the Party A channel executed	r
lastdata	String (80)	The application data for the last application the Party A channel executed	r
start	Date/time	The time the CDR was created	r
answer	Date/time	The time when Party A was answered, or when the bridge between Party A and Party B was created	r
end	Date/time	The time when the CDR was finished. This occurs when either party hangs up, or when the bridge between the parties is broken	r
duration	Integer	The time in seconds from start until end	r
billsec	Integer	The time in seconds from answer until end	r
disposition	Enum	The final known disposition of the CDR record. See CDR dispositions for possible values	r
amaflags	Enum	A flag specified on the Party A channel. See AMA records for possible values	r/w

userfield	String (255)	A user defined field set on the channels. If set on both the Party A and Party B channel, the userfields of both are concatenated and separated by a ;	r/w
uniqueid	String (32)	A unique identifier for the Party A channel	r
linkedid	String (32)	A unique identifier that unites multiple CDR records. See linkedid propagation for more details	r
peeraccount	String (80)	The account code of the Party B channel	r/w
sequence	Integer	A numeric value that, combined with uniqueid and linkedid, can be used to uniquely identify a single CDR record	r

Any of the values may be accessed using the [CDR function](#). Any value that is read/write may be modified using this same function. CDR field values cannot be modified once the CDR is finalized.

Dispositions

Dispositions represent the final state of the call from the perspective of Party A.

Value	Description	Hangup Cause Mapping	Dial Status Mapping
NO ANSWER	The channel was never answered. This is the default disposition for an unanswered channel.	Any not explicitly listed	<ul style="list-style-type: none"> CANCEL NOANSWER
CONGESTION	The channel dialed something that was congested.	<ul style="list-style-type: none"> AST_CAUSE_CONGESTION 	CONGESTION
FAILED	<p>The channel attempted to dial but the call failed.</p> <div>  <p>The congestion setting in <code>cdr.conf</code> can result in the <code>AST_CAUSE_CONGESTION</code> hang up cause or the <code>CONGESTION</code> dial status to map to this disposition.</p> </div>	<ul style="list-style-type: none"> AST_CAUSE_CONGESTION AST_CAUSE_NO_ROUTE_DESTINATION AST_CAUSE_UNREGISTERED 	<ul style="list-style-type: none"> CONGESTION FAILED
BUSY	The channel attempted to dial but the remote party was busy.	<ul style="list-style-type: none"> AST_CAUSE_BUSY 	<ul style="list-style-type: none"> BUSY
ANSWERED	The channel was answered. When the channel is answered, the hangup cause no longer changes the disposition.	Any not explicitly listed	<ul style="list-style-type: none"> ANSWER

AMA Flags

AMA Flags are set on a channel and are conveyed in the CDR. They inform billing systems how to treat the particular CDR. Asterisk provides no additional semantics regarding these flags - they are present simply to help external systems classify CDRs.

- OMIT
- BILLING

- DOCUMENTATION

User Defined Fields

Any CDR record may have user defined fields associated with it. Fields can be added to either the Party A or Party B channel. Note that not all CDR backends support user defined fields; in those cases the user field is simply dropped when the CDR is dispatched to the backend.

If a Party A channel and a Party B channel both contain a field with the same key, only the Party A channel's field will be written to the CDR.

User defined CDR fields are created using the [CDR function](#), and read using the same function.

Scenarios



The following scenarios show examples of CDRs created in common use cases. If a particular scenario is not shown below, the CDRs created during the scenario should still match the behavior described previously. Some applications, however, may have undefined behavior as their use is not common or the mechanism by which they manipulate channels does not allow for the capturing of the channel state.

Undefined behavior means that the behavior of CDRs in those cases is unsupported and will not be addressed as a bug.

Unanswered "Inbound" Call



Unanswered calls may not always be logged to CDR backends if the configuration has explicitly disabled unanswered calls.

Alice calls into Asterisk at extension 500 using a SIP phone and, during dialplan execution of a NoOp(), hangs up.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice <100>"	100	500	default	SIP/alice-00000000		NoOp		2013-03-04 13:11:18		2013-03-04 13:11:20	2	0	NO ANSWER	DOCUMENTATION	100		Asterisk-01-1362424276.2		34	Asterisk-01-1362424276.2

Unanswered "Outbound" Call

Asterisk creates a call file to dial Alice and playback tt-monkeys to her. Alice, anticipating the screeching of howler monkeys, never picks up the phone.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
		s	default	SIP/alice-00000000		AppDial	SIP/Alice	2013-03-04 13:11:18		2013-03-04 13:11:20	2	0	NO ANSWER	DOCUMENTATION			Asterisk-01-1362424276.2		4	Asterisk-01-1362424276.2

Single Party

Alice calls into Asterisk's VoiceMailMain application. This implicitly Answers the channel. She checks her voicemail for awhile, then hangs up.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice<100>"	100	8500	default	SIP/alice-00000000		Hangup		2013-03-04 13:11:18	2013-03-04 13:11:20	2013-03-04 13:12:18	60	58	ANSWERED	DOCUMENTATION	100		Asterisk-01-1362424276.2		112	Asterisk-01-1362424276.2

Basic Two Party Calls

Two party calls can be initiated in a variety of ways. Several of the more common ways are illustrated here.

Basic Call

Alice calls into Asterisk, which dials Bob. Bob Answers, and a bridge is formed between Alice and Bob. Alice and Bob talk for awhile, then Bob hangs up. This breaks the bridge between Alice and Bob, and Alice is hung up on as well.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice<100>"	100	200	default	SIP/alice-00000000	SIP/bob-00000001	Dial	SIP/bob,Tt	2013-03-04 13:11:18	2013-03-04 13:11:26	2013-03-04 13:13:18	120	112	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		12	Asterisk-01-1362424276.2

Unanswered Dial

Alice calls into Asterisk, which dials Bob. Bob refuses to pick up his phone, and the call eventually times out.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice<100>"	100	200	default	SIP/alice-00000000	SIP/bob-00000001	Dial	SIP/bob,10,Tt	2013-03-04 13:11:18		2013-03-04 13:11:28	10	0	NO ANSWER	DOCUMENTATION			Asterisk-01-1362424276.2		1	Asterisk-01-1362424276.2

Parallel Dial

Alice calls into Asterisk, which dials Bob's SIP desk phone as well as his IAX2 soft phone. Both ring for awhile, and Bob eventually presses the Answer button on his IAX2 soft phone.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
------	-----	-----	----------	---------	------------	---------	----------	-------	--------	-----	----------	-------	-------------	----------	-------------	-------------	----------	-----------	----------	--------

"Alice<100>"	100	200	default	SIP/alice-0000000	SIP/bob-0000001	Dial	SIP/bob&IA X2/bob,,Tt	2013-03-04 13:11:18		2013-03-04 13:11:28	10	0	NO ANSWER	DOCUMENTATION			Asterisk-01-1362424276.2		12	Asterisk-01-1362424276.2
"Alice<100>"	100	200	default	SIP/alice-0000000	IAX2/bob-0000000	Dial	SIP/bob&IA X2/bob,,Tt	2013-03-04 13:11:18	2013-03-04 13:11:28	2013-03-04 13:12:28	70	60	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		13	Asterisk-01-1362424276.2

Call Forward

Alice calls into Asterisk, which dials Bob's SIP desk phone. Bob is on vacation, and the SIP phone returns a "302" and redirects Asterisk to dial his SIP mobile.

Transfers

Transfers create multiple CDRs. In general, a CDR is created for each path of communication between two endpoints. Note that Asterisk does **not** attempt to compute the total duration or billing time of any of the various channels involved - it is up to the businesses consuming CDRs to know whether or not the amount of time they want to bill a party includes the transfer, the time spent dialing another party, consultation time, etc.

Blind Transfer

Alice calls into Asterisk, which dials Bob's SIP phone. Bob answers, and Alice and Bob talk for awhile. Eventually, Bob decides to send Alice off to Charlie, and he blind transfers Alice to Charlie's extension. Asterisk dials Charlie's SIP phone, and Charlie answers. Alice and Charlie talk for awhile until Alice decides to hang up.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice<100>"	100	200	default	SIP/alice-0000000	SIP/bob-0000001	Dial	SIP/bob,,Tt	2013-03-04 13:11:18	2013-03-04 13:11:28	2013-03-04 13:11:48	30	20	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		101	Asterisk-01-1362424276.2
"Alice<100>"	100	300	default	SIP/alice-0000000	SIP/charlie-0000002	Dial	SIP/charlie,,Tt	2013-03-04 13:11:48	2013-03-04 13:11:53	2013-03-04 13:12:53	65	60	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		102	Asterisk-01-1362424276.2

Attended Transfer to Channel

Alice calls into Asterisk, which dials Bob's SIP phone. Bob answers, and Alice and Bob talk for awhile. Eventually, Bob decides to send Alice off to Charlie, and he initiates an attended transfer. Alice is put on hold, and Bob dials Charlie's extension. Asterisk dials Charlie's SIP phone, and Charlie answers. Bob and Charlie talk for a bit, and Charlie agrees to talk to Alice. Bob completes the attended transfer, Alice is taken off hold, and Alice and Charlie are bridged. Alice talks to Charlie for awhile, then hangs up.

clid	src	dst	dco ntex t	cha nnel	dstc han nel	last app	last data	start	ans wer	end	dur atio n	bills ec	disp ositi on	ama flag s	acc oun tcode	peer acc ount	uniq ueid	user field	seq uen ce	link edid
"Alice <100>"	100	200	default	SIP/ alice- 000000	SIP/ bob- 000001	Dial	SIP/ bob, ,Tt	201 3-03 -04 13:1 1:18	201 3-03 -04 13:1 1:28	201 3-03 -04 13:1 2:18	60	50	ANS WER RED	DO CU ME NTA TION			Asterisk- 01-1 362 424 276. 2		101	Asterisk- 01-1 362 424 276. 2
"Bob <200>"	200	300	default	SIP/ bob- 000001	SIP/ charlie- 000002	Dial	SIP/ charlie,, Tt	201 3-03 -04 13:1 1:48	201 3-03 -04 13:1 1:53	201 3-03 -04 13:1 2:18	30	26	ANS WER RED	DO CU ME NTA TION			Asterisk- 01-1 362 424 280. 1		102	Asterisk- 01-1 362 424 276. 2
"Alice <100>"	100	300	default	SIP/ alice- 000000	SIP/ charlie- 000002	Dial	SIP/ bob, ,Tt	201 3-03 -04 13:1 2:18	201 3-03 -04 13:1 2:18	201 3-03 -04 13:1 2:53	45	45	ANS WER RED	DO CU ME NTA TION			Asterisk- 01-1 362 424 276. 2		103	Asterisk- 01-1 362 424 276. 2



In the example above, note the following:

- Hold time is not reflected in CDRs.
- When Bob dials Charlie, he becomes the Party A channel. However, the `linkedid` from Alice 'wins', and so the CDR reflects the `linkedid` from Alice's CDR.
- Alice and Charlie are bridged automatically by the attended transfer, so their start and answer times are identical.
- The billsec/duration of Alice and Charlie are reflective of their portion of the call, and do not include the times from Alice and Bob.

Attended Transfer to Application

Alice calls into Asterisk, which dials Bob's SIP phone. Bob answers, and Alice and Bob talk for awhile. Eventually, Bob decides to send Alice off into Charlie's voicemail mailbox, and he initiates an attended transfer. Alice is put on hold, and Bob dials an extension that calls into VoiceMail. Bob enters in the Charlie's voicemail mailbox number, then completes the attended transfer to put Alice into the voicemail mailbox. Alice records some voicemail, then hangs up.

clid	src	dst	dco ntex t	cha nnel	dstc han nel	last app	last data	start	ans wer	end	dur atio n	bills ec	disp ositi on	ama flag s	acc oun tcode	peer acc ount	uniq ueid	user field	seq uen ce	link edid
"Alice <100>"	100	200	default	SIP/ alice- 000000	SIP/ bob- 000001	Dial	SIP/ bob, ,Tt	201 3-03 -04 13:1 1:18	201 3-03 -04 13:1 1:28	201 3-03 -04 13:1 2:18	60	50	ANS WER RED	DO CU ME NTA TION			Asterisk- 01-1 362 424 276. 2		101	Asterisk- 01-1 362 424 276. 2

"Bob <200>"	200	8500	default	SIP/ bob-0000001		Voicemail	300	201-03-04 13:11:48	201-03-04 13:11:49	201-03-04 13:12:18	30	29	ANSWERED	DOCUMENTATION			Asterisk-01-1362424280.1		102	Asterisk-01-1362424276.2
"Alice <100>"	100	8500	default	SIP/ alice-0000000		Voicemail	300	201-03-04 13:12:18	201-03-04 13:12:18	201-03-04 13:13:18	60	60	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		103	Asterisk-01-1362424276.2

Blonde Transfer

Alice calls into Asterisk, which dials Bob's SIP phone. Bob answers, and Alice and Bob talk for awhile. Eventually, Bob decides to send Alice off to Charlie, and he initiates an attended transfer. Alice is put on hold, and Bob dials Charlie's extension. Asterisk dials Charlie's SIP phone, but before Charlie answers Bob hangs up. Asterisk recognizes that this is a blonde transfer, takes Alice off hold, and ties Charlie's ringing phone to Alice. Charlie answers, Alice talks to Charlie for awhile, then hangs up.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice <100>"	100	200	default	SIP/ alice-0000000	SIP/ bob-0000001	Dial	SIP/ bob,Tt	201-03-04 13:11:18	201-03-04 13:11:28	201-03-04 13:12:18	60	50	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		101	Asterisk-01-1362424276.2
"Bob <200>"	200	300	default	SIP/ bob-0000001	SIP/ charlie-0000002	Dial	SIP/ charlie,,Tt	201-03-04 13:11:48		201-03-04 13:11:50	2	0	NO ANSWER	DOCUMENTATION			Asterisk-01-1362424280.1		102	Asterisk-01-1362424276.2
"Alice <100>"	100	300	default	SIP/ alice-0000000	SIP/ charlie-0000002	Dial	SIP/ charlie,,Tt	201-03-04 13:11:50	201-03-04 13:11:55	201-03-04 13:12:30	40	35	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		103	Asterisk-01-1362424276.2

Three Way Call

Alice calls into Asterisk, which dials Bob's SIP phone. Bob answers, and Alice and Bob talk for awhile. Eventually, Bob decides to bring Charlie into the mix. He puts Alice on hold and dials Charlie's extension. Asterisk dials Charlie's SIP phone, and Charlie answers. Bob and Charlie talk for awhile, and Bob then finishes the three-way call by finalizing the attempt. Alice is taken off hold, and Alice, Bob, and Charlie can all talk. Eventually, Bob hangs up, and all parties are ejected and hung up on.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
------	-----	-----	----------	---------	------------	---------	----------	-------	--------	-----	----------	-------	-------------	----------	-------------	-------------	----------	-----------	----------	--------

"Alice <100>"	100	200	default	SIP/alice-0000000	SIP/bob-0000001	Dial	SIP/bob,,Tt	2013-03-04 13:11:18	2013-03-04 13:11:28	2013-03-04 13:13:18	120	110	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		101	Asterisk-01-1362424276.2
"Bob <200>"	200	300	default	SIP/bob-0000001	SIP/charlie-0000002	Dial	SIP/charlie,,Tt	2013-03-04 13:11:48	2013-03-04 13:11:53	2013-03-04 13:12:08	20	15	ANSWERED	DOCUMENTATION			Asterisk-01-1362424280.1		102	Asterisk-01-1362424276.2
"Alice <100>"	100	300	default	SIP/alice-0000000	SIP/charlie-0000002	Dial	SIP/bob,,Tt	2013-03-04 13:12:08	2013-03-04 13:12:08	2013-03-04 13:13:18	70	70	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		103	Asterisk-01-1362424276.2
"Bob <200>"	200	300	default	SIP/bob-0000001	SIP/charlie-0000002	Dial	SIP/charlie,,Tt	2013-03-04 13:12:08	2013-03-04 13:12:08	2013-03-04 13:13:18	70	70	ANSWERED	DOCUMENTATION			Asterisk-01-1362424280.1		104	Asterisk-01-1362424276.2



In the example above, note the following:

- The consultation between Bob and Charlie is treated as a separate conversation from the conversation between all three parties. Thus, there are two CDRs between Bob and Charlie.
- Because the path of communication never was broken between Alice and Bob (despite Alice being put on hold), there is only one CDR for Alice to Bob.

SIP Attended Transfer

Alice calls into Asterisk, which dials Bob's SIP phone. Bob answers, and Alice and Bob talk for awhile. Eventually, Alice decides to transfer Bob to Charlie, and performs an attended transfer using her SIP phone. Bob is put on hold. Alice and Charlie talk for awhile, and then Alice finishes the attended transfer. Bob is taken off hold, and bridged with Charlie. Alice is hung up.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice <100>"	100	200	default	SIP/alice-0000000	SIP/bob-0000001	Dial	SIP/bob	2013-03-04 13:11:18	2013-03-04 13:11:28	2013-03-04 13:12:18	60	50	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		101	Asterisk-01-1362424276.2

"Bob <200>"	200	300	default	SIP/ bob- 000 000 02	SIP/ char lie-0 000 000 3	Dial	SIP/ char lie	201 3-03 -04 13:1 1:48	201 3-03 -04 13:1 1:53	201 3-03 -04 13:1 2:08	20	15	ANS WER	DO CU ME NTA TION			Aste risk- 01-1 362 424 280. 1		102	Aste risk- 01-1 362 424 276. 2
"Alice <100>"	100	200	default	SIP/ alice -000 000 00	SIP/ char lie-0 000 000 3	Dial	SIP/ bob	201 3-03 -04 13:1 2:18	201 3-03 -04 13:1 2:18	201 3-03 -04 13:1 2:28	10	10	ANS WER	DO CU ME NTA TION			Aste risk- 01-1 362 424 276. 2		103	Aste risk- 01-1 362 424 276. 2



The important point to note here is that a SIP attended transfer uses two channels to communicate with Bob - SIP/bob-00000001 and SIP/bob-00000002. The CDR records are associated by virtue of the `linkedid` field.

Local Channels

Local channels are a special Asterisk construct that create a path of communication between two bridges or a bridge and an application. A Local channel always consists of two channels with the same name prefix - the first of the Local channel pair is delineated with a `;1`; the second is delineated with a `;2`. Local channels have two different modes in Asterisk:

- They can exist permanently. In that case, the Local channel pair appears as two separate channels. From the perspective of CDRs, they are treated as such with an implicit bridge between the channels. CDRs with a permanent Local channel pair will share the same `linkedid`.
- They can optimize. A Local channel optimization occurs when a pair of Local channels exist between two bridges and the Local channel has been configured to optimize. If that situation occurs, the Local channel will cause the two bridges to merge into a single merge, and the Local channel will disappear.

Each situation and how it appears in CDRs is explored further below.

Non-optimizing Local Channels

Local channel to an application

An external application [Originates](#) a Local channel. The first half of the Local channel Dials Alice over a SIP channel. The second half of the Local channel is placed into her VoiceMail account. Alice listens to her VoiceMail through the Local channel, then hangs up.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkedid
""	dial_alice	100	default	Local/dial_alice@default-00000001;1	SIP/alice-000002	Dial	SIP/alice,,t	201 3-03 -04 13:1 1:18	201 3-03 -04 13:1 1:28	201 3-03 -04 13:1 2:18	60	50	ANS WER	DO CU ME NTA TION			Aste risk- 01-1 362 424 290. 1		101	Aste risk- 01-1 362 424 290. 1

""	voicemail		default	Local/dial_alice@default-0000001;2		VoiceMailMain	100	2013-03-04 13:11:28	2013-03-04 13:11:28	2013-03-04 13:12:18	50	50	ANSWERED	DOCUMENTATION			Asterisk-01-1362424290.1		102	Asterisk-01-1362424290.1
----	-----------	--	---------	------------------------------------	--	---------------	-----	---------------------	---------------------	---------------------	----	----	----------	---------------	--	--	--------------------------	--	-----	--------------------------

Local channel between bridges

An external application [Originates](#) a Local channel. The first half of the Local channel Dials Alice over a SIP channel; Alice answers. This triggers the second half of the Local channel, which Dials Bob. Bob Answers, and Alice and Bob talk. Alice hangs up, the Local channels are hung up, and Bob is hung up on.

clid	src	dst	context	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
""	dial_alice	100	default	Local/dial_alice@default-0000001;1	SIP/alice-00002	Dial	SIP/alice,,t	2013-03-04 13:11:18	2013-03-04 13:11:28	2013-03-04 13:12:18	60	50	ANSWERED	DOCUMENTATION			Asterisk-01-1362424290.1		101	Asterisk-01-1362424290.1
""	dial_bob	200	default	Local/dial_alice@default-0000001;2	SIP/bob-00003	Dial	SIP/bob,,t	2013-03-04 13:11:28	2013-03-04 13:11:38	2013-03-04 13:12:18	50	40	ANSWERED	DOCUMENTATION			Asterisk-01-1362424290.1		102	Asterisk-01-1362424290.1

Local Channel Optimization

When a Local channel optimization occurs, the CDR records associated with the Local channel are finalized. New CDR records are generated for the channels in the merged bridge, per the rules in [CDR Consolidation](#).



CDR properties set on optimized Local channels are **not** propagated to other channels. Setting CDR information on optimizing Local channels will cause that information to be lost.

In prior versions of Asterisk it was sometimes necessary to set CDR information on Local channels - with the addition of [Pre-Dial handlers](#), it is always possible to set CDR information on the appropriate channel at the time of creation.

Call Hold

Call Hold is a state of the media between two or more channels, and not a change in the actual bridging of those channels. As such, Call Hold is **not** reflected in CDRs. Channels may be put on hold, taken off hold, put on hold again, forgotten about, found again, taken off hold, etc. without affecting CDRs at all.

Call Park

Call Park is different from Call Hold. Whereas Call Hold is a change of media state, Call Park implies that the channel has been moved into a state where it can be retrieved by any other channel. As such, calls in Park receive their own CDR.

Alice calls into Asterisk and Bob answers. Alice says she wants to talk to Charlie, but Bob isn't sure Charlie wants to talk to Alice so he blind transfers her

into Park. Alice sits, waiting in her parking slot, listening to serenading robots while Bob asks if Charlie wants to talk to Alice. Charlie says sure, so he picks Alice up out of Park and they talk for awhile before Alice hangs up.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice<100>"	100	200	default	SIP/alice-00000000	SIP/bob-00000001	Dial	SIP/bob,Tt	2013-03-04 13:11:18	2013-03-04 13:11:28	2013-03-04 13:11:38	20	10	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		101	Asterisk-01-1362424276.2
"Alice<100>"	100	700	default	SIP/alice-00000000		Park	6000, default,200,1	2013-03-04 13:11:38	2013-03-04 13:11:38	2013-03-04 13:12:38	60	60	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		102	Asterisk-01-1362424276.2
"Alice<100>"	100	300	default	SIP/alice-00000000	SIP/charlie-00000002	Park	6000, default,200,1	2013-03-04 13:12:38	2013-03-04 2:38	2013-03-04 3:38	60	60	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		103	Asterisk-01-1362424276.2

Call Queues

Call Queue - Example 1

Alice calls into Asterisk and enters [Queue](#) without being Answered. She waits in the queue for a period of time. At some point in time, she enters into the head of the queue and the queue performs a ring-all on the members of the queue. There are two queue members - Bob and Charlie. Bob is out to lunch, so his queue member is paused and it returns a Busy indication. Charlie, on the other hand, is not busy and answers. Alice and Charlie talk for awhile, and eventually Alice hangs up.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice<100>"	100	800	default	SIP/alice-00000000	SIP/bob-00000001	Queue	complaints	2013-03-04 13:11:18		2013-03-04 13:14:18	180	0	BUSY	DOCUMENTATION			Asterisk-01-1362424276.2		101	Asterisk-01-1362424276.2
"Alice<100>"	100	800	default	SIP/alice-00000000	SIP/charlie-00000002	Queue	complaints	2013-03-04 13:11:18	2013-03-04 2:18	2013-03-04 4:18	180	120	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		102	Asterisk-01-1362424276.2

Call Queue - Example 2

Alice calls into Asterisk and enters Queue without being Answered. She waits in the queue for a period of time. At some point in time, she enters into the

head of the queue and the queue performs a round-robin strategy ring on the members of the queue. Bob is rung first, but being lazy, he ignores his phone. After some time of ringing, it times out and goes on to Charlie. When Charlie's SIP phone rings, he immediately answers. Alice and Charlie talk for some period of time, and eventually Alice hangs up.

clid	src	dst	dco ntext	chan nel	dstc han nel	last app	last data	start	ans wer	end	dur atio n	bills ec	disp ositi on	ama flag s	acc oun tcode	peer acc ount	uniq ueid	user field	seq uen ce	link edid
"Alice <100>"	100	800	default	SIP/alice-00000001	SIP/bob-00000001	Queue	complaints	2013-03-04 13:11:18		2013-03-04 13:14:18	180	0	NO ANSWER	DOCUMENTATION			Asterisk-01-1362424276.2		101	Asterisk-01-1362424276.2
"Alice <100>"	100	800	default	SIP/alice-000000002	SIP/charlie-00000002	Queue	complaints	2013-03-04 13:11:18	2013-03-04 13:12:38	2013-03-04 13:14:18	180	100	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		102	Asterisk-01-1362424276.2

Conference Call

Alice calls into Asterisk and joins a ConfBridge conference. Bob does a bit later as well. Finally, Charlie joins the Conference. After talking for awhile, Bob realizes he's late for lunch and hangs up. Alice and Charlie talk for a bit longer, then finally Alice hangs up. Charlie stays in the conference for another second before he hangs up as well.

clid	src	dst	dco ntext	chan nel	dstc han nel	last app	last data	start	ans wer	end	dur atio n	bills ec	disp ositi on	ama flag s	acc oun tcode	peer acc ount	uniq ueid	user field	seq uen ce	link edid
"Alice <100>"	100	1000	default	SIP/alice-00000001	SIP/bob-00000001	ConfBridge	1000,public_bridge,public_user,public_menu	2013-03-04 13:11:18	2013-03-04 13:11:28	2013-03-04 13:14:18	180	170	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		1	Asterisk-01-1362424276.2
"Bob <200>"	200	1000	default	SIP/bob-00000001	SIP/charlie-00000002	ConfBridge	1000,public_bridge,public_user,public_menu	2013-03-04 13:11:28	2013-03-04 13:12:18	2013-03-04 13:14:18	170	120	ANSWERED	DOCUMENTATION			Asterisk-01-1362424280.1		2	Asterisk-01-1362424276.2

"Alice<100>"	100	1000	default	SIP/alice-0000000	SIP/charlie-00000002	Conf Bridge	1000,public_bridged,public_user,public_mnu	2013-03-04 13:11:18	2013-03-04 13:12:18	2013-03-04 13:15:18	240	180	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2	3	Asterisk-01-1362424276.2
--------------	-----	------	---------	-------------------	----------------------	-------------	--	---------------------	---------------------	---------------------	-----	-----	----------	---------------	--	--	--------------------------	---	--------------------------

A Complex Example

Alice calls into Asterisk and Dials Bob. Bob answers, and he and Alice talk for awhile. Bob realizes that what Alice really needs is to talk to Sales, so he blind transfers Alice off to the Sales Queue. As Alice is heading off to the Sales Queue, Bob realizes that he should talk with Charlie about Alice, so he Dials Charlie and he and Charlie talk for awhile. Alice enters into the Sales Queue, where she waits for a bit while agents David and Frank are dialed using Local channels to SIP devices. Alice is eventually Answered by David, a sales agent. David and Alice talk for a bit, but David isn't able to sell her on their new fantastic product, so he puts Alice on hold for a bit and calls Ellen from engineering. Ellen agrees to be on the call, and Alice, David, and Ellen are put into a three-way call. Around this time, Charlie decides that he should talk to Alice as well. He transfers himself to the Sales bridge, hanging up on Bob in the process. This turns the Sales bridge into a four-way call. The four parties talk for awhile, and eventually Alice is sold on the new whiz-bang product, so she hangs up. Ellen realizes she isn't need any more either, and hangs up as well. Charlie and David talk about the weather for awhile, and then Charlie hangs up, hanging up David as well.

- Alice calls into Asterisk and Dials Bob. Bob and Alice talk for awhile.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice<100>"	100	200	default	SIP/alice-0000000	SIP/bob-00000001	Dial	SIP/bob,Tt	2013-03-04 13:00:00	2013-03-04 13:00:05	2013-03-04 13:01:00	60	55	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		1	Asterisk-01-1362424276.2

- Bob realizes Alice wants to talk to Sales, so he blind transfers her off to the Sales Queue. Alice enters into the Sales Queue, where she waits for a bit while agents David and Frank are dialed using Local channels to SIP devices. Alice is eventually Answered by David, a sales agent.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Alice<100>"	100	700	default	SIP/alice-0000000	Local/member1@default-00000001;1	Queue	sales	2013-03-04 13:01:00		2013-03-04 13:02:00	60	0	NO ANSWER	DOCUMENTATION			Asterisk-01-1362424276.2		2	Asterisk-01-1362424276.2

	700	member1	default	Local/mer1@default-0000001;2	SIP/frank-000002	Dial	SIP/frank	2013-03-04 13:01:50		2013-03-04 13:02:00	10	0	NO ANSWER	DOCUMENTATION			Asterisk-01-1362424280.1		3	Asterisk-01-1362424276.2
"Alice <100>"	100	700	default	SIP/alice-0000000@default-0000002;1	Local/mer2@default-0000002;1	Queue	sales	2013-03-04 13:01:00	2013-03-04 13:02:00	2013-03-04 13:05:00	240	180	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		4	Asterisk-01-1362424276.2
	700	member2	default	Local/mer2@default-0000002;2	SIP/david-000003	Dial	SIP/david	2013-03-04 13:02:00	2013-03-04 13:02:05	2013-03-04 13:06:00	240	235	ANSWERED	DOCUMENTATION			Asterisk-01-1362424280.3		5	Asterisk-01-1362424276.2



Note that with non-optimizing Local channels, the duration of the Alice to the Local channel (which in turns passes media to/from David) may not reflect the length of time that the Local channel to David is in the bridge. As we'll see, additional channels joining the bridge will change that CDR's durations.

- Meanwhile, Bob calls Charlie.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
"Bob <200>"	200	300	default	SIP/bob-0000004	SIP/charlie-0000005	Dial	SIP/charlie,,Tt	2013-03-04 13:01:10	2013-03-04 13:01:20	2013-03-04 13:03:20	130	120	ANSWERED	DOCUMENTATION			Asterisk-01-1362424277.1		6	Asterisk-01-1362424277.1

- David and Alice talk for a bit, but David isn't able to sell her on their new fantastic product, so he puts Alice on hold for a bit and calls Ellen from engineering. Ellen agrees to be on the call, and Alice, David, and Ellen are put into a three-way call.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	bills	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkid
------	-----	-----	----------	---------	------------	---------	----------	-------	--------	-----	----------	-------	-------------	----------	-------------	-------------	----------	-----------	----------	--------

"David <900>"	900	901	default	SIP/david-000003	SIP/ellen-000006	Dial	SIP/ellen,,Tt	2013-03-04 13:02:30	2013-03-04 13:02:40	2013-03-04 13:03:00	30	20	ANSWERED	DOCUMENTATION			Asterisk-01-1362424300.1	7	Asterisk-01-1362424276.2
"Alice <100>"	100	700	default	SIP/alice-0000000	SIP/ellen-000006	Queue	sales	2013-03-04 13:03:00	2013-03-04 13:03:00	2013-03-04 13:05:00	120	120	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2	8	Asterisk-01-1362424276.2
	700	member2	default	Local/member2@default-00000002;2	SIP/ellen-000006	Dial	SIP/david	2013-03-04 13:03:00	2013-03-04 13:03:00	2013-03-04 13:05:00	120	120	ANSWERED	DOCUMENTATION			Asterisk-01-1362424280.3	9	Asterisk-01-1362424276.2



During the consultation period, David's SIP channel directly Dials Ellen's SIP device. However, when Ellen joins the bridge with David, it is the Local channel to David that is in the bridge, not David's SIP channel. Thus, the CDR reflects the Local channel to Ellen's SIP channel.

- Around this time, Charlie decides that he should talk to Alice as well. He transfers himself to the Sales bridge, hanging up on Bob in the process. This turns the Sales bridge into a four-way call. The four parties talk for awhile, and eventually Alice is sold on the new whiz-bang product, so she hangs up. Ellen realizes she isn't need any more either, and hangs up as well. Charlie and David talk about the weather for awhile, and then Charlie hangs up, hanging up David as well.

clid	src	dst	dcontext	channel	dstchannel	lastapp	lastdata	start	answer	end	duration	billed	disposition	amaflags	accountcode	peeraccount	uniqueid	userfield	sequence	linkedid
"Alice <100>"	100	700	default	SIP/alice-0000000	SIP/charlie-00000005	Queue	sales	2013-03-04 13:03:20	2013-03-04 13:03:20	2013-03-04 13:05:00	100	100	ANSWERED	DOCUMENTATION			Asterisk-01-1362424276.2		10	Asterisk-01-1362424276.2
"Charlie <300>"	300	701	default	SIP/charlie-00000005	Local/member2@default-00000002;2	Bridge	SIP/alice-0000000	2013-03-04 13:03:20	2013-03-04 13:03:20	2013-03-04 13:06:00	160	160	ANSWERED	DOCUMENTATION			Asterisk-01-1362424278.1		11	Asterisk-01-1362424276.2

"Charlie <300>"	300	701	default	SIP/charlie-0000005	SIP/ellen-000006	Bridge	SIP/alice-000000	2013-03-04 13:03:20	2013-03-04 13:03:20	2013-03-04 13:05:05	105	105	ANSWERED	DOCUMENTATION			Asterisk-01-13624278.1	12	Asterisk-01-13624276.2
--------------------	-----	-----	---------	---------------------	------------------	--------	------------------	---------------------	---------------------	---------------------	-----	-----	----------	---------------	--	--	------------------------	----	------------------------



Because Charlie's channel is older than either the Local channel to David's SIP channel or Ellen's SIP channel, Charlie is chosen as Party A for those CDRs. Alice, on the other hand, is older than Charlie, so she is Party A for that CDR. Because Alice is the oldest channel, her `linkedid` is propagated to all CDRs in the bridge. However, the CDR between Charlie and Bob is not affected, as Bob is the Party A in that CDR and the CDR would already have been dispatched by the time Charlie joined this bridge.

Asterisk CDR APIs

The following details high level APIs that Asterisk provides for manipulating CDRs.



These still describe applications/functions available in Asterisk 11. When documentation has been updated for these applications/functions for Asterisk 12, the links will be updated appropriately.

Applications

NoCDR

When this application is executed on a channel, the channel is no longer considered for CDRs. Any previous CDRs involving the channel will continue to be updated.



This application is deprecated. It is now recommended to use the [CDR_PROP](#) function instead.

ForkCDR

ForkCDR now does significantly less than it used to. The application will finalize the current CDR and create a new CDR for the party A channel. The new CDR record may or may not inherit properties of the previously finalized CDR, based on parameters passed to the application.

ResetCDR

ResetCDR has two purposes:

1. It resets the start time/answer time for the current CDR. If the channel is answered, the start time and answer time will reflect when ResetCDR was called on the channel.
2. Alternatively, it simply enables CDRs on a channel that previously had NoCDR executed on it.

Functions

CDR

Retrieve or modify a field in a CDR record.

CDR_PROP

Modify a fundamental property on a CDR record.

Example: Setting the called party as Party A of the CDR


```
[default]

exten => 100,1,NoOp()
same =>      n,Dial(SIP/bob,,b(default^callee_handler^1))
same =>      n,Hangup()

exten => callee_handler,1,NoOp()
same =>      n,Set(CDR(party_A)=true)
same =>      n,Return()
```

Configuration

The following parameters can be configured for the CDR engine. Additional CDR backends may have their own configuration settings that are outside the scope of this specification.

General Settings

The following settings must appear in the context `general`.

Name	Type	Description
enable	Boolean	Enable/disable the CDR engine
batch	Boolean	Dispatch CDRs in batches.
unanswered	Boolean	Dispatch unanswered CDRs. See Definition of Unanswered for more information.
congestion	Boolean	Treat congestion calls as failed calls
endbeforehexten	Boolean	Finalize CDRs before the <code>h</code> extension or hangup handlers are executed
initiatedseconds	Boolean	Count microseconds for the purposes of the <code>billsec</code> field
size	Integer	The number of records to buffer before initiating a batch
time	Integer	The time, in seconds, before initiating a batch
scheduleronly	Boolean	Deprecated. See <code>usethreadpool</code> instead.
usethreadpool	Boolean	For any CDRs that are dispatched, use a thread pool thread to perform the dispatching. This prevents the CDR taskprocessor thread from being blocked by any CDR backends.
safeshutdown	Boolean	Block Asterisk shutdown on dispatching of CDRs

Asterisk 12 CEL Specification

- Introduction
 - Scope
 - Terminology
- CEL Overview
- Record Types
 - Stand-Alone Records
 - Channel Start
 - Channel End
 - Answer
 - Hangup
 - Application Start
 - Application End
 - User Defined
 - Linked ID End
 - Interaction Records
 - Bridge Enter
 - Bridge Exit
 - Forward
 - Park Start
 - Park End
 - Pickup
 - Meta-Records
 - Blind Transfer
 - Attended Transfer
 - Bridge-Bridge Attended Transfers
 - Bridge-App Attended Transfers
 - App-App Attended Transfers
 - Local Channel Optimization
 - Removed Records
- Record Fields
 - Primary Fields
 - CallerID Name
 - CallerID Number
 - CallerID ANI
 - CallerID RDNIS
 - CallerID DNID
 - Extension
 - Context
 - Channel Name
 - Application Name
 - Application Data
 - Account Code
 - Peer Account Code
 - Unique ID
 - Linked ID
 - AMA Flags
 - Record Type Specific Fields
 - User Defined Name
 - Extra
- Logging Backends
 - Custom
 - Manager
 - ODBC
 - PGSQL
 - RADIUS
 - SQLite
 - TDS

- [Example Scenarios](#)
 - [Two-Participant Bridge](#)
 - [Multi-participant Conference](#)
 - [Dial Nominal](#)
 - [Dial Busy](#)
 - [Blind Transfer](#)
 - [Attended Transfer](#)

Introduction

Channel Event Logging (CEL) provides a series of records describing the state of channels in Asterisk to any of several event recording backends. CEL records provide substantially more information than CDRs and thus allow an Asterisk User to construct their own more complex billing system.

As a result of the bridging work done for Asterisk 12, CEL behavior has changed for several events that occur in the system. The most significant changes are:

- AST_CEL_BRIDGE_ENTER and AST_CEL_BRIDGE_EXIT have been introduced to denote participant changes in bridges.
- AST_CEL_BRIDGE_START and AST_CEL_BRIDGE_END have been removed as they no longer applies to the new bridging framework.
- AST_CEL_BRIDGE_UPDATE has been removed as it no longer applies to the new bridging framework.
- AST_CEL_LOCAL_OPTIMIZE has been added to describe local channel optimizations that occur.
- All linkedid accounting and record generation is now handled within the CEL engine.
- The peer field is only used in BRIDGE_ENTER and BRIDGE_EXIT records.

Scope

This CEL specification applies to Asterisk 12. While some portions of this specification are applicable to prior versions of Asterisk, other portions are specific to Asterisk 12 and their counterparts in prior versions are not discussed.

Terminology

Term	Definition
CEL	Channel Event Logging. The focus of this documentation.
CEL record	An individual event record produced by the CEL engine.
CDR	Call Detail Record. An alternative method of extracting billing information from Asterisk. Simpler, but less flexible.
Stasis	The internal message bus in Asterisk that conveys state to the CEL engine.
Primary	The channel around which a CEL record is focused.
AMI	Asterisk Manager Interface
CSV	Comma Separated Values. A format commonly used for tabular data when stored outside of a database.

CEL Overview

A CEL record contains information about a system event including a partial dump of the Primary's state and may contain data relevant to that specific record type such as channel names, bridge unique identifiers, channel variable values, or other miscellaneous information. The CEL engine tracks changes in individual channel state and guarantees ordering of records for a given Primary, but does not guarantee ordering of records in relation to other Primaries. The exception to this record ordering occurs with meta-records which occur adjacent to the events they describe. Applicable event ordering is provided in the descriptions below. CEL output does not describe interaction with MeetMe conferences other than MeetMe as an application.

Record Types

The records produced by the CEL engine can be grouped in to three general categories:

Stand-Alone Records

These records convey a channel event on the channel that does not involve channels or bridges other than the Primary.

Channel Start

An AST_CEL_CHANNEL_START record is generated when a channel is created. This record introduces a new Primary and is the first record available for all Primaries.

Channel End

An AST_CEL_CHAN_END record is generated when a channel is destroyed. This record indicates that a Primary is going away and that there will be no further records for this Primary with the exception of AST_CEL_LINKEDID_END.

Answer

An AST_CEL_ANSWER record is generated when a channel is answered. Depending on the state transitions that occur on a Primary, this record may not be generated.

Hangup

An AST_CEL_HANGUP record is generated when a channel is hung up. This record will occur on every Primary prior to channel destruction.

Application Start

An AST_CEL_APP_START record is generated when a channel enters an application. This record will always be generated before its corresponding AST_CEL_APP_END.

Application End

An AST_CEL_APP_END record is generated when a channel exits an application. This record will be generated after its corresponding AST_CEL_APP_START, but is not guaranteed to be generated on hangup.

User Defined

An AST_CEL_USER_DEFINED record is generated when a channel enters the CELGenUserEvent application. The application sets the user defined name field and additional information in the extra field in the "extra" key.

Linked ID End

An AST_CEL_LINKEDID_END record is generated when the last channel using the given linked ID is destroyed or the last instance of a linked ID is overwritten by a different linked ID. This is the only type of record that may occur after AST_CEL_CHANNEL_END.

Interaction Records

These records convey the Primary's interactions with other channels or bridges.

Bridge Enter

An AST_CEL_BRIDGE_ENTER record is generated when a channel enters a bridge. The entering channel is the Primary for this event. Additional information is conveyed in the extra field under the "bridge_id" key. All other channels in the bridge at the time of entry are available in the peer field as a comma-separated list.

Bridge Exit

An AST_CEL_BRIDGE_EXIT record is generated when a channel exits a bridge. The leaving channel is the Primary for this event. Additional information is conveyed in the extra field under the "bridge_id" key. All other channels in the bridge at the time of exit are available in the peer field as a comma-separated list.

Forward

An AST_CEL_FORWARD record is generated when a dialing channel is forwarded elsewhere by a dialed channel. The dialing channel is the Primary for

this event. Additional information is conveyed in the extra field under the "forward" key.

Park Start

An AST_CEL_PARK_START record is generated when a channel is parked. The parked channel is the Primary for this event. Additional information is conveyed in the extra field under the keys "parker_dial_string" and "parking_lot".

Park End

An AST_CEL_PARK_START record is generated when a channel is unparked. The unparked channel is the Primary for this event. Additional information is conveyed in the extra field under the "reason" key. This record always occurs after its corresponding AST_CEL_PARK_START.

Pickup

An AST_CEL_PICKUP record is generated when a channel is picked up. The picked up channel (also known as the target) is the Primary for this record. The name of the channel that is picking up is conveyed in the extra field under the "pickup_channel" key.

Meta-Records

These records convey additional context relating to surrounding CEL records

Blind Transfer

An AST_CEL_BLINDTRANSFER record is generated when a blind transfer feature is activated on a bridge. The initiating channel is the Primary for this record. Additional information is conveyed in the extra field under the "extension", "context", and "bridge_id" keys.

Attended Transfer

An AST_CEL_ATTENDEDTRANSFER record is generated when an attended transfer is successfully performed.

Bridge-Bridge Attended Transfers

This type of attended transfer occurs when both involved channels are bridged. The initiating channel is the Primary for this record. Additional information is conveyed in the extra field under the "bridge1_id", "channel2_name", and "bridge2_id" keys.

The records associated with this type of transfer will vary depending on the configuration of the bridges involved and the number of channels involved. Possible methods of accomplishing the transfer include (but are not limited to) channel swap, bridge merge, and bridge link via a local channel.

Bridge-App Attended Transfers

This type of attended transfer occurs when one involved channel is bridged while the other is running an application. The bridged channel is the Primary for this record. Additional information is conveyed in the extra field under the "bridge1_id", "channel2_name", and "app" keys.

App-App Attended Transfers

Attended transfers involving only channels that are running applications are not currently possible. This is not possible with internal transfers since there is no bridge involved to handle the feature codes and any externally initiated attended transfer that attempts to bridge two app-bound channels will fail.

Local Channel Optimization

An AST_CEL_LOCAL_OPTIMIZE record is generated when a local channel optimization attempt completes successfully. The semi-one (local channel ending in ';1') channel is the Primary for this event. The name of the semi-two (local channel ending in ';2') channel is conveyed in the extra field under the "local_two" key.

Removed Records

The following record types are no longer available in Asterisk 12:

- AST_CEL_BRIDGE_START
- AST_CEL_BRIDGE_END
- AST_CEL_CONF_START
- AST_CEL_CONF_END
- AST_CEL_CONF_ENTER

- AST_CEL_CONF_EXIT
- AST_CEL_HOOKFLASH
- AST_CEL_3WAY_START
- AST_CEL_3WAY_END
- AST_CEL_BRIDGE_UPDATE
- AST_CEL_TRANSFER

Record Fields

Primary Fields

These fields are populated exclusively from their corresponding fields on the Primary in a consistent manner for every CEL record.

CallerID Name

The name identifying the caller for this channel.

CallerID Number

The number identifying the caller for this channel.

CallerID ANI

Automatic Number Identification caller information provided for this channel.

CallerID RDNIS

Redirecting information for this channel.

CallerID DNID

Dialed Number Identification for this channel.

Extension

The extension in which this channel is currently executing.

Context

The context in which this channel is currently executing.

Channel Name

The name of this channel.

Application Name

The name of the application that this channel is currently executing.

Application Data

The data provided to the application being executed.

Account Code

The account code used for billing.

Peer Account Code

The peer channel's account code.

Unique ID

This channel's instance unique identifier.

Linked ID

This channel's current linked ID which is affected by bridging operations. This identifier starts as the channel's unique ID.

AMA Flags

This channel's Automated Message Accounting flags.

Record Type Specific Fields

These fields vary or may be blank depending on the CEL record type.

User Defined Name

This field is only used for `AST_CEL_USER_DEFINED` and conveys the user-specified event type.

Extra

This field contains a JSON blob describing additional record-type-specific information.

Logging Backends

CEL provides several methods of logging records to be processed at a later time. CEL only publishes record types to backends that are enabled in the general CEL configuration. Sample configurations are provided with the Asterisk 12 source for all of these backends.

Custom

The Custom CEL output module provides logging capability to a CSV file in a format described in the configuration file. This module is configured in `cel_custom.conf`.

Manager

The manager CEL output module publishes records over AMI as CEL events with the record type published under the "EventName" key. This module is configured in `cel.conf` in the `[manager]` section.

ODBC

The ODBC CEL output module provides logging capability to any ODBC-compatible database. This module is configured in `cel_odbc.conf`.

PGSQL

The PGSQL CEL output module provides logging capability to PostgreSQL databases when it is desirable to avoid the ODBC abstraction layer. This module is configured in `cel_pgsql.conf`.

RADIUS

The RADIUS CEL output module allows the CEL engine to publish records to a RADIUS server. This module is configured in `cel.conf` in the `[radius]` section.

SQLite

The SQLite CEL output module provides logging capability to a SQLite3 database in a format described in its configuration file. This module is configured in `cel_sqlite3_custom.conf`.

TDS

The TDS CEL output module provides logging capability to Sybase or Microsoft SQL Server databases when it is desirable to avoid the ODBC abstraction layer. This module is configured in `cel_tds.conf`.

Example Scenarios

For the following scenarios, assume the CEL engine is configured to generate the following record types:

- `AST_CEL_CHANNEL_START`
- `AST_CEL_CHAN_END`
- `AST_CEL_BRIDGE_ENTER`
- `AST_CEL_BRIDGE_EXIT`

Two-Participant Bridge

The following scenario demonstrates channel creation, channel destruction, bridge start, and bridge end:

Event	Record	Primary	Extra
Channel Alice is created	<code>AST_CEL_CHANNEL_START</code>	Alice	
Channel Bob is created	<code>AST_CEL_CHANNEL_START</code>	Bob	
Bridge Link is created			
Alice enters bridge Link	<code>AST_CEL_BRIDGE_ENTER</code>	Alice	<code>{"bridge_id": "Link"}</code>
Bob enters bridge Link	<code>AST_CEL_BRIDGE_ENTER</code>	Bob	<code>{"bridge_id": "Link"}</code>
Bob exits bridge Link	<code>AST_CEL_BRIDGE_EXIT</code>	Bob	<code>{"bridge_id": "Link"}</code>
Bob is destroyed	<code>AST_CEL_CHAN_END</code>	Bob	
Alice exits bridge Link	<code>AST_CEL_BRIDGE_EXIT</code>	Alice	<code>{"bridge_id": "Link"}</code>
Alice is destroyed	<code>AST_CEL_CHAN_END</code>	Alice	

Multi-participant Conference

The following scenario demonstrates conversion of a bridge to a multi-participant conference:

Event	Record	Primary	Extra
Channel Alice is created	<code>AST_CEL_CHANNEL_START</code>	Alice	
Channel Bob is created	<code>AST_CEL_CHANNEL_START</code>	Bob	
Channel Charlie is created	<code>AST_CEL_CHANNEL_START</code>	Charlie	
Channel David is created	<code>AST_CEL_CHANNEL_START</code>	David	
Bridge Link is created			
Alice enters bridge Link	<code>AST_CEL_CONF_ENTER</code>	Alice	<code>{"bridge_id", "Link"}</code>
Bob enters bridge Link	<code>AST_CEL_CONF_ENTER</code>	Bob	<code>{"bridge_id", "Link"}</code>
Charlie enters bridge Link	<code>AST_CEL_CONF_ENTER</code>	Charlie	<code>{"bridge_id", "Link"}</code>
David enters bridge Link	<code>AST_CEL_CONF_ENTER</code>	David	<code>{"bridge_id", "Link"}</code>
Alice exits bridge Link	<code>AST_CEL_CONF_EXIT</code>	Alice	<code>{"bridge_id", "Link"}</code>

Alice is destroyed	AST_CEL_CHAN_END	Alice	
Bob exits bridge Link	AST_CEL_CONF_EXIT	Bob	{"bridge_id", "Link"}
Bob is destroyed	AST_CEL_CHAN_END	Bob	
Charlie exits bridge Link	AST_CEL_CONF_EXIT	Charlie	{"bridge_id", "Link"}
Charlie is destroyed	AST_CEL_CHAN_END	Charlie	
David exits bridge Link	AST_CEL_CONF_EXIT	David	{"bridge_id", "Link"}
David is destroyed	AST_CEL_CHAN_END	David	

Dial Nominal

For this scenario, assume that AST_CEL_ANSWER, AST_CEL_HANGUP, AST_CEL_APP_START, and AST_CEL_APP_END are configured in addition to the aforementioned record types and that "Dial" is configured to be watched.

The following scenario demonstrates a Dial that results in an answer followed by bridging and hangup:

Event	Record	Primary	Extra
Channel Alice is created	AST_CEL_CHANNEL_START	Alice	
Alice executes Dial(SIP/Bob)	AST_CEL_APP_START	Alice	
Channel Bob is created	AST_CEL_CHANNEL_START	Bob	
Bob answers	AST_CEL_ANSWER	Bob	
Alice answers	AST_CEL_ANSWER	Alice	
Bridge Link is created			
Alice enters bridge Link	AST_CEL_BRIDGE_ENTER	Alice	{"bridge_id": "Link"}
Bob enters bridge Link	AST_CEL_BRIDGE_ENTER	Bob	{"bridge_id": "Link"}
Bob initiates hangup, exits bridge Link	AST_CEL_BRIDGE_EXIT	Bob	{"bridge_id": "Link"}
Bob completes hang up	AST_CEL_HANGUP	Bob	{"hangupcause":16,"dialstatus":"","hangupsource":"Bob"}
Bob is destroyed	AST_CEL_CHAN_END	Bob	
Alice exits bridge Link	AST_CEL_BRIDGE_EXIT	Alice	{"bridge_id": "Link"}
Alice is hung up	AST_CEL_HANGUP	Alice	{"hangupcause":16,"dialstatus":"ANSWER","hangupsource":""}
Alice is destroyed	AST_CEL_CHAN_END	Alice	

Dial Busy

For this scenario, assume that AST_CEL_ANSWER, AST_CEL_HANGUP, AST_CEL_APP_START, and AST_CEL_APP_END are configured in addition to the aforementioned record types and that "Dial" is configured to be watched. The following scenario demonstrates a Dial that results in a busy:

Event	Record	Primary	Extra
Channel Alice is created	AST_CEL_CHANNEL_START	Alice	
Alice executes Dial(SIP/Bob)	AST_CEL_APP_START	Alice	

Channel Bob is created	AST_CEL_CHANNEL_START	Bob	
Bob responds BUSY	AST_CEL_HANGUP	Bob	{"hangupcause":21,"dialstatus":"","hangupsource":""}
Bob is destroyed	AST_CEL_CHAN_END	Bob	
Alice is hung up	AST_CEL_HANGUP	Alice	{"hangupcause":17,"dialstatus":"BUSY","hangupsource":""}
Alice is destroyed	AST_CEL_CHAN_END	Alice	

Blind Transfer

For this scenario, assume that AST_CEL_HANGUP is configured in addition to the aforementioned record types. The following scenario demonstrates a blind transfer:

Event	Record	Primary	Extra
Channel Alice is created	AST_CEL_CHANNEL_START	Alice	
Channel Bob is created	AST_CEL_CHANNEL_START	Bob	
Alice answers	AST_CEL_ANSWER	Alice	
Bob answers	AST_CEL_ANSWER	Bob	
Bridge Link is created			
Bob enters bridge Link	AST_CEL_BRIDGE_ENTER	Bob	{"bridge_id":"Link"}
Alice enters bridge Link	AST_CEL_BRIDGE_ENTER	Alice	{"bridge_id":"Link"}
Alice initiates a blind transfer to exten@context	AST_CEL_BLINDTRANSFER	Alice	{"bridge_id":"Link","extension":"exten","context":"context"}
Alice exits bridge Link	AST_CEL_BRIDGE_EXIT	Alice	{"bridge_id":"Link"}
Alice is hung up	AST_CEL_HANGUP	Alice	{"hangupcause":16,"dialstatus":"","hangupsource":""}
Alice is destroyed	AST_CEL_CHANNEL_END	Alice	
A local channel pair is created to handle dialplan	AST_CEL_CHANNEL_START	Local1	
	AST_CEL_CHANNEL_START	Local2	
Local1 enters bridge Link	AST_CEL_BRIDGE_ENTER	Local1	{"bridge_id":"Link"}
Local2 executes dialplan at exten@context			
Local2 is eventually hung up by the dialplan	AST_CEL_HANGUP	Local2	{"hangupcause":16,"dialstatus":"","hangupsource":""}
Hangup is initiated on Local1, exiting bridge Link	AST_CEL_BRIDGE_EXIT	Local1	{"bridge_id":"Link"}
Local1 is hung up	AST_CEL_HANGUP	Local1	{"hangupcause":16,"dialstatus":"","hangupsource":""}
Local1 is destroyed	AST_CEL_CHANNEL_END	Local1	

Local2 is destroyed	AST_CEL_CHANNEL_END	Local2	
Bob is the last channel and so is hung up	AST_CEL_HANGUP	Bob	{"hangupcause":16,"dialstatus":"","hangupsource":""}
Bob is destroyed	AST_CEL_CHANNEL_END	Bob	

Attended Transfer

For this scenario, assume that AST_CEL_ANSWER and AST_CEL_HANGUP are configured in addition to the aforementioned record types. The following scenario demonstrates a channel-swapping attended transfer:

Event	Record	Primary	Extra
Channel Alice is created	AST_CEL_CHANNEL_START	Alice	
Channel Bob is created	AST_CEL_CHANNEL_START	Bob	
Alice answers	AST_CEL_ANSWER	Alice	
Bob answers	AST_CEL_ANSWER	Bob	
Bridge Link1 is created			
Bob enters bridge Link1	AST_CEL_BRIDGE_ENTER	Bob	{"bridge_id":"Link1"}
Alice enters bridge Link1	AST_CEL_BRIDGE_ENTER	Alice	{"bridge_id":"Link1"}
Channel Charlie is created	AST_CEL_CHANNEL_START	Charlie	
Channel David is created	AST_CEL_CHANNEL_START	David	
Charlie answers	AST_CEL_ANSWER	Charlie	
David answers	AST_CEL_ANSWER	David	
Bridge Link2 is created			
David enters bridge Link2	AST_CEL_BRIDGE_ENTER	Bob	{"bridge_id":"Link2"}
Charlie enters bridge Link2	AST_CEL_BRIDGE_ENTER	Alice	{"bridge_id":"Link2"}
An attended transfer between Alice and David begins			
Bob exits bridge Link1	AST_CEL_BRIDGE_EXIT	Bob	{"bridge_id":"Link1"}
Bob enters bridge Link2	AST_CEL_BRIDGE_ENTER	Bob	{"bridge_id":"Link2"}
David exits bridge Link2	AST_CEL_BRIDGE_EXIT	David	{"bridge_id":"Link2"}
David is hung up	AST_CEL_HANGUP	David	{"hangupcause":16,"dialstatus":"","hangupsource":""}
David is destroyed	AST_CEL_CHANNEL_END	David	
Alice and David execute an attended transfer	AST_CEL_ATTENDEDTRANSFER	Alice	{"bridge1_id":"Link1","channel2_name":"David","bridge2_id":"Link2"}
Alice exits bridge Link1	AST_CEL_BRIDGE_EXIT	Alice	{"bridge_id":"Link1"}
Alice is hung up	AST_CEL_HANGUP	Alice	{"hangupcause":16,"dialstatus":"","hangupsource":""}
Alice is destroyed	AST_CEL_CHANNEL_END	Alice	

Bob exits bridge Link2	AST_CEL_BRIDGE_EXIT	Bob	{"bridge_id":"Link2"}
Charlie exits bridge Link2	AST_CEL_BRIDGE_EXIT	Charlie	{"bridge_id":"Link2"}
Bob is hung up	AST_CEL_HANGUP	Bob	{"hangupcause":16,"dialstatus":"","hangupsource":""}
Bob is destroyed	AST_CEL_CHANNEL_END	Bob	
Charlie is hung up	AST_CEL_HANGUP	Charlie	{"hangupcause":16,"dialstatus":"","hangupsource":""}
Charlie is destroyed	AST_CEL_CHANNEL_END	Charlie	

Note that the ATTENDEDTRANSFER event does not necessarily occur before or after the records it is related to.

Asterisk 12 Command Reference

This page is the top level page for the XML/JSON derived documentation in Asterisk 12:

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

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

Asterisk 12 AGI Commands

Asterisk 12 AGICommand_answer

ANSWER

Synopsis

Answer channel

Description

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

Syntax

ANSWER

Arguments

See Also

- [Asterisk 12 AGICommand_hangup](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_asyncagi break

ASYNACGI BREAK

Synopsis

Interrupts Async AGI

Description

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

Syntax

```
ASYNACGI BREAK
```

Arguments

See Also

- [Asterisk 12 AGICommand_hangup](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_channel status

CHANNEL STATUS

Synopsis

Returns status of the connected channel.

Description

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

Return values:

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

Syntax

```
CHANNEL STATUS CHANNELNAME
```

Arguments

- `channelname`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_control stream file

CONTROL STREAM FILE

Synopsis

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

Description

Send the given file, allowing playback to be controlled by the given digits, if any. Use double quotes for the digits if you wish none to be permitted. If `offsetms` is provided then the audio will seek to `offsetms` before play starts. Returns 0 if playback completes without a digit being pressed, or the ASCII numerical value of the digit if one was pressed, or -1 on error or if the channel was disconnected. Returns the position where playback was terminated as `endpos`.

It sets the following channel variables upon completion:

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_database get

DATABASE GET

Synopsis

Gets database value

Description

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

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

Example return code: 200 result=1 (testvariable)

Syntax

```
DATABASE GET FAMILY KEY
```

Arguments

- family
- key

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_exec

EXEC

Synopsis

Executes a given Application

Description

Executes *application* with given *options*.

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

Syntax

```
EXEC APPLICATION OPTIONS
```

Arguments

- application
- options

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_get full variable

GET FULL VARIABLE

Synopsis

Evaluates a channel expression

Description

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

Example return code: 200 result=1 (testvariable)

Syntax

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

Arguments

- `variablename`
- `channel name`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_get option

GET OPTION

Synopsis

Stream file, prompt for DTMF, with timeout.

Description

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

Syntax

```
GET OPTION FILENAME ESCAPE_DIGITS TIMEOUT
```

Arguments

- filename
- escape_digits
- timeout

See Also

- [Asterisk 12 AGICommand_stream file](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_get variable

GET VARIABLE

Synopsis

Gets a channel variable.

Description

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

Example return code: 200 result=1 (testvariable)

Syntax

```
GET VARIABLE VARIABLENAME
```

Arguments

- *variablename*

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_gosub

GOSUB

Synopsis

Cause the channel to execute the specified dialplan subroutine.

Description

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

Syntax

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

Arguments

- context
- extension
- priority
- optional-argument

See Also

- [Asterisk 12 Application_GoSub](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_noop

NOOP

Synopsis

Does nothing.

Description

Does nothing.

Syntax

```
NOOP
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_record file

RECORD FILE

Synopsis

Records to a given file.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_say time

SAY TIME

Synopsis

Says a given time.

Description

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

Syntax

```
SAY TIME TIME ESCAPE_DIGITS
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_send text

SEND TEXT

Synopsis

Sends text to channels supporting it.

Description

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

Syntax

```
SEND TEXT TEXT TO SEND
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_set autohangup

SET AUTOHANGUP

Synopsis

Autohangup channel in some time.

Description

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

Syntax

```
SET AUTOHANGUP TIME
```

Arguments

- *time*

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_set callerid

SET CALLERID

Synopsis

Sets callerid for the current channel.

Description

Changes the callerid of the current channel.

Syntax

```
SET CALLERID NUMBER
```

Arguments

- number

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_set music

SET MUSIC

Synopsis

Enable/Disable Music on hold generator

Description

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

Always returns 0.

Syntax

```
SET MUSIC CLASS
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_set variable

SET VARIABLE

Synopsis

Sets a channel variable.

Description

Sets a variable to the current channel.

Syntax

```
SET VARIABLE VARIABLENAME VALUE
```

Arguments

- variablename
- value

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_speech create

SPEECH CREATE

Synopsis

Creates a speech object.

Description

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

Syntax

```
SPEECH CREATE ENGINE
```

Arguments

- engine

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_speech deactivate grammar

SPEECH DEACTIVATE GRAMMAR

Synopsis

Deactivates a grammar.

Description

Deactivates the specified grammar on the speech object.

Syntax

```
SPEECH DEACTIVATE GRAMMAR GRAMMAR NAME
```

Arguments

- grammar name

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_speech destroy

SPEECH DESTROY

Synopsis

Destroys a speech object.

Description

Destroy the speech object created by `SPEECH CREATE`.

Syntax

```
SPEECH DESTROY
```

Arguments

See Also

- [Asterisk 12 AGICommand_speech create](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_speech load grammar

SPEECH LOAD GRAMMAR

Synopsis

Loads a grammar.

Description

Loads the specified grammar as the specified name.

Syntax

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

Arguments

- grammar name
- path to grammar

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_speech recognize

SPEECH RECOGNIZE

Synopsis

Recognizes speech.

Description

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

Syntax

SPEECH RECOGNIZE PROMPT TIMEOUT OFFSET
--

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_speech set

SPEECH SET

Synopsis

Sets a speech engine setting.

Description

Set an engine-specific setting.

Syntax

```
SPEECH SET NAME VALUE
```

Arguments

- name
- value

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_stream file

STREAM FILE

Synopsis

Sends audio file on channel.

Description

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

It sets the following channel variables upon completion:

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

Syntax

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

Arguments

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

See Also

- [Asterisk 12 AGICommand_control stream file](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 AGICommand_verbose

VERBOSE

Synopsis

Logs a message to the asterisk verbose log.

Description

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

Syntax

```
VERBOSE MESSAGE LEVEL
```

Arguments

- message
- level

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AGICommand_wait for digit

WAIT FOR DIGIT

Synopsis

Waits for a digit to be pressed.

Description

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

Syntax

```
WAIT FOR DIGIT TIMEOUT
```

Arguments

- `timeout`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AMI Actions

Asterisk 12 ManagerAction_AbsoluteTimeout

AbsoluteTimeout

Synopsis

Set absolute timeout.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_Agents

Agents

Synopsis

Lists agents and their status.

Description

Will list info about all defined agents.

Syntax

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

Arguments

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

See Also

- [Asterisk 12 ManagerEvent_Agents](#)
- [Asterisk 12 ManagerEvent_AgentsComplete](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_AGI

AGI

Synopsis

Add an AGI command to execute by Async AGI.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_AOCMessage

AOCMessage

Synopsis

Generate an Advice of Charge message on a channel.

Description

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

Syntax

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

Arguments

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

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_Atxf

Atxfer

Synopsis

Attended transfer.

Description

Attended transfer.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_BindTransfer

BindTransfer

Synopsis

Bind transfer channel(s) to the given destination

Description

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

Syntax

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

Arguments

- Channel
- Context
- Exten

See Also

- [Asterisk 12 ManagerAction_Redirect](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_Bridge

Bridge

Synopsis

Bridge two channels already in the PBX.

Description

Bridge together two channels already in the PBX.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_BridgeDestroy

BridgeDestroy

Synopsis

Destroy a bridge.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402438

Asterisk 12 ManagerAction_BridgeInfo

BridgeInfo

Synopsis

Get information about a bridge.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_BridgeKick

BridgeKick

Synopsis

Kick a channel from a bridge.

Description

The channel is removed from the bridge.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402438

Asterisk 12 ManagerAction_BridgeList

BridgeList

Synopsis

Get a list of bridges in the system.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_BridgeTechnologyList

BridgeTechnologyList

Synopsis

List available bridging technologies and their statuses.

Description

Returns detailed information about the available bridging technologies.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402438

Asterisk 12 ManagerAction_BridgeTechnologySuspend

BridgeTechnologySuspend

Synopsis

Suspend a bridging technology.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402438

Asterisk 12 ManagerAction_BridgeTechnologyUnsuspend

BridgeTechnologyUnsuspend

Synopsis

Unsuspend a bridging technology.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402438

Asterisk 12 ManagerAction_Challenge

Challenge

Synopsis

Generate Challenge for MD5 Auth.

Description

Generate a challenge for MD5 authentication.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_ChangeMonitor

ChangeMonitor

Synopsis

Change monitoring filename of a channel.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_Command

Command

Synopsis

Execute Asterisk CLI Command.

Description

Run a CLI command.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_ConfbridgeList

ConfbridgeList

Synopsis

List participants in a conference.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_ConfbridgeListRooms

ConfbridgeListRooms

Synopsis

List active conferences.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_ConfbridgeLock

ConfbridgeLock

Synopsis

Lock a Confbridge conference.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_ConfbridgeSetSingleVideoSrc

ConfbridgeSetSingleVideoSrc

Synopsis

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

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_ConfbridgeStartRecord

ConfbridgeStartRecord

Synopsis

Start recording a Confbridge conference.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_ConfbridgeStopRecord

ConfbridgeStopRecord

Synopsis

Stop recording a Confbridge conference.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_ConfbridgeUnlock

ConfbridgeUnlock

Synopsis

Unlock a Confbridge conference.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_ControlPlayback

ControlPlayback

Synopsis

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

Description

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



Note

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

Syntax

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

Arguments

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



Note

The default *skipms* value is 3000 ms.

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



Note

The default *skipms* value is 3000 ms.

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

See Also

- [Asterisk 12 Application_Playback](#)
- [Asterisk 12 Application_ControlPlayback](#)
- [Asterisk 12 AGICommand_stream file](#)
- [Asterisk 12 AGICommand_control stream file](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_CoreSettings

CoreSettings

Synopsis

Show PBX core settings (version etc).

Description

Query for Core PBX settings.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_CreateConfig

CreateConfig

Synopsis

Creates an empty file in the configuration directory.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_DAHDI DialOffhook

DAHDI DialOffhook

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 Unknown

Asterisk 12 ManagerAction_DAHDIIDNDoff

DAHDIIDNDoff

Synopsis

Toggle DAHDI channel Do Not Disturb status OFF.

Description

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



Note

Feature only supported by analog channels.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_DAHDIIDNDon

DAHDIIDNDon

Synopsis

Toggle DAHDI channel Do Not Disturb status ON.

Description

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



Note

Feature only supported by analog channels.

Syntax

```
Action: DAHDIIDNDon
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 Unknown

Asterisk 12 ManagerAction_DAHDIHangup

DAHDIHangup

Synopsis

Hangup DAHDI Channel.

Description

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



Note

Valid only for analog channels.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_DAHDIRestart

DAHDIRestart

Synopsis

Fully Restart DAHDI channels (terminates calls).

Description

Equivalent to the CLI command "dahdi restart".

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_DAHDIShowChannels

DAHDIShowChannels

Synopsis

Show status of DAHDI channels.

Description

Similar to the CLI command "dahdi show channels".

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_DAHDITransfer

DAHDITransfer

Synopsis

Transfer DAHDI Channel.

Description

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



Note

Valid only for analog channels.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_DBPut

DBPut

Synopsis

Put DB entry.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_Filter

Filter

Synopsis

Dynamically add filters for the current manager session.

Description

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

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

Syntax

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

Arguments

- **ActionID** - ActionID for this transaction. Will be returned.
- **Operation**
 - **Add** - Add a filter.
- **Filter** - Filters can be whitelist or blacklist
Example whitelist filter: "Event: Newchannel"
Example blacklist filter: "!Channel: DAHDI.*"
This filter option is used to whitelist or blacklist events per user to be reported with regular expressions and are allowed if both the regex matches and the user has read access as defined in manager.conf. Filters are assumed to be for whitelisting unless preceeded by an exclamation point, which marks it as being black. Evaluation of the filters is as follows:
 - If no filters are configured all events are reported as normal.
 - If there are white filters only: implied black all filter processed first, then white filters.
 - If there are black filters only: implied white all filter processed first, then black filters.
 - If there are both white and black filters: implied black all filter processed first, then white filters, and lastly black filters.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_FilterList

FilterList

Synopsis

Show current event filters for this session

Description

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

Syntax

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_GetConfigJSON

GetConfigJSON

Synopsis

Retrieve configuration (JSON format).

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_Getvar

Getvar

Synopsis

Gets a channel variable.

Description

Get the value of a global or local channel variable.



Note

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_Hangup

Hangup

Synopsis

Hangup channel.

Description

Hangup a channel.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_IAXnetstats

IAXnetstats

Synopsis

Show IAX Netstats.

Description

Show IAX channels network statistics.

Syntax

Action: IAXnetstats

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_IAXpeerlist

IAXpeerlist

Synopsis

List IAX Peers.

Description

List all the IAX peers.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_IAXpeers

IAXpeers

Synopsis

List IAX peers.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_IAXregistry

IAXregistry

Synopsis

Show IAX registrations.

Description

Show IAX registrations.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_JabberSend_res_jabber

JabberSend - [res_jabber]

Synopsis

Sends a message to a Jabber Client.

Description

Sends a message to a Jabber Client.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_JabberSend_res_xmpp

JabberSend - [res_xmpp]

Synopsis

Sends a message to a Jabber Client.

Description

Sends a message to a Jabber Client.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_ListCommands

ListCommands

Synopsis

List available manager commands.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_LocalOptimizeAway

LocalOptimizeAway

Synopsis

Optimize away a local channel when possible.

Description

A local channel created with "/" 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_MailboxStatus

MailboxStatus

Synopsis

Check mailbox.

Description

Checks a voicemail account for status.

Returns whether there are messages waiting.

Message: Mailbox Status.

Mailbox: *mailboxid*.

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_MeetmeList

MeetmeList

Synopsis

List participants in a conference.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_MeetmeMute

MeetmeMute

Synopsis

Mute a Meetme user.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_MessageSend

MessageSend

Synopsis

Send an out of call message to an endpoint.

Description

Syntax

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

Arguments

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

Technology: PJSIP

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

Technology: SIP

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

Technology: XMPP

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

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

Technology: PJSIP

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

Technology: SIP

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

Technology: XMPP

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

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403796

Asterisk 12 ManagerAction_MixMonitor

MixMonitor

Synopsis

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

Description

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

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_ModuleLoad

ModuleLoad

Synopsis

Module management.

Description

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

Syntax

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

Arguments

- **ActionID** - ActionID for this transaction. Will be returned.
- **Module** - Asterisk module name (including .so extension) or subsystem identifier:
 - cdr
 - dnsmgr
 - extconfig
 - enum
 - acl
 - manager
 - http
 - logger
 - features
 - dsp
 - udptl
 - indications
 - cel
 - plc
- **LoadType** - The operation to be done on module. Subsystem identifiers may only be reloaded.
 - load
 - unload
 - reload

If no module is specified for a `reload` loadtype, all modules are reloaded.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_MuteAudio

MuteAudio

Synopsis

Mute an audio stream.

Description

Mute an incoming or outgoing audio stream on a channel.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_Originate

Originate

Synopsis

Originate a call.

Description

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

Syntax

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

Arguments

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

See Also

- [Asterisk 12 ManagerEvent_OriginateResponse](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_Park

Park

Synopsis

Park a channel.

Description

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

Syntax

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

Arguments

- **ActionID** - ActionID for this transaction. Will be returned.
- **Channel** - Channel name to park.
- **TimeoutChannel** - Channel name to use when constructing the dial string that will be dialed if the parked channel times out.
- **Timeout** - Overrides the timeout of the parking lot for this park action. Specified in milliseconds, but will be converted to seconds. Use a value of 0 to nullify the timeout.
- **Parkinglot** - The parking lot to use when parking the channel

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_ParkedCalls

ParkedCalls

Synopsis

List parked calls.

Description

List parked calls.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_Parkinglots

Parkinglots

Synopsis

Get a list of parking lots

Description

List all parking lots as a series of AMI events

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_PauseMonitor

PauseMonitor

Synopsis

Pause monitoring of a channel.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_Ping

Ping

Synopsis

Keepalive command.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_PJSIPNotify

PJSIPNotify

Synopsis

Send a NOTIFY to an endpoint.

Description

Send a NOTIFY to an endpoint.

Parameters will be placed into the notify as SIP headers.

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Endpoint` - The endpoint to which to send the NOTIFY.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_PJSIPQualify

PJSIPQualify

Synopsis

Qualify a chan_pjsip endpoint.

Description

Qualify a chan_pjsip endpoint.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_PJSIPShowEndpoint

PJSIPShowEndpoint

Synopsis

Detail listing of an endpoint and its objects.

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403134

Asterisk 12 ManagerAction_PJSIPShowEndpoints

PJSIPShowEndpoints

Synopsis

Lists PJSIP endpoints.

Description

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

Syntax

Action: PJSIPShowEndpoints

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403134

Asterisk 12 ManagerAction_PJSIPShowRegistrationsInbound

PJSIPShowRegistrationsInbound

Synopsis

Lists PJSIP inbound registrations.

Description

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

Syntax

Action: PJSIPShowRegistrationsInbound

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403134

Asterisk 12 ManagerAction_PJSIPShowRegistrationsOutbound

PJSIPShowRegistrationsOutbound

Synopsis

Lists PJSIP outbound registrations.

Description

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

Syntax

Action: PJSIPShowRegistrationsOutbound

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403134

Asterisk 12 ManagerAction_PJSIPShowSubscriptionsInbound

PJSIPShowSubscriptionsInbound

Synopsis

Lists subscriptions.

Description

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

Syntax

Action: PJSIPShowSubscriptionsInbound

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403134

Asterisk 12 ManagerAction_PJSIPShowSubscriptionsOutbound

PJSIPShowSubscriptionsOutbound

Synopsis

Lists subscriptions.

Description

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

Syntax

Action: PJSIPShowSubscriptionsOutbound
--

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403134

Asterisk 12 ManagerAction_PJSIPUnregister

PJSIPUnregister

Synopsis

Unregister an outbound registration.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_PlayDTMF

PlayDTMF

Synopsis

Play DTMF signal on a specific channel.

Description

Plays a dtmf digit on the specified channel.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_PresenceState

PresenceState

Synopsis

Check Presence State

Description

Report the presence state for the given presence provider.

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_PRIShowSpans

PRIShowSpans

Synopsis

Show status of PRI spans.

Description

Similar to the CLI command "pri show spans".

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_QueueAdd

QueueAdd

Synopsis

Add interface to queue.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_QueueLog

QueueLog

Synopsis

Adds custom entry in queue_log.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_QueueMemberRingInUse

QueueMemberRingInUse

Synopsis

Set the ringinuse value for a queue member.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_QueuePause

QueuePause

Synopsis

Makes a queue member temporarily unavailable.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_QueueReset

QueueReset

Synopsis

Reset queue statistics.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_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 Unknown

Asterisk 12 ManagerAction_Queues

Queues

Synopsis

Queues.

Description

Syntax

Action: Queues

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_Reload

Reload

Synopsis

Send a reload event.

Description

Send a reload event.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_Setvar

Setvar

Synopsis

Set a channel variable.

Description

Set a global or local channel variable.



Note

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_ShowDialPlan

ShowDialPlan

Synopsis

Show dialplan contexts and extensions

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_SIPnotify

SIPnotify

Synopsis

Send a SIP notify.

Description

Sends a SIP Notify event.

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_SIPpeers

SIPpeers

Synopsis

List SIP peers (text format).

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_SIPpeerstatus

SIPpeerstatus

Synopsis

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

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_SIPqualifypeer

SIPqualifypeer

Synopsis

Qualify SIP peers.

Description

Qualify a SIP peer.

Syntax

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

Arguments

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

See Also

- [Asterisk 12 ManagerEvent_SIPQualifyPeerDone](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_SIPshowpeer

SIPshowpeer

Synopsis

show SIP peer (text format).

Description

Show one SIP peer with details on current status.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_SIPshowregistry

SIPshowregistry

Synopsis

Show SIP registrations (text format).

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_SKINNYdevices

SKINNYdevices

Synopsis

List SKINNY devices (text format).

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_SKINNYlines

SKINNYlines

Synopsis

List SKINNY lines (text format).

Description

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_SKINNYshowdevice

SKINNYshowdevice

Synopsis

Show SKINNY device (text format).

Description

Show one SKINNY device with details on current status.

Syntax

```
Action: SKINNYshowdevice  
ActionID: <value>  
Device: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_StopMixMonitor

StopMixMonitor

Synopsis

Stop recording a call through MixMonitor, and free the recording's file handle.

Description

This action stops the audio recording that was started with the `MixMonitor` action on the current channel.

Syntax

```
Action: StopMixMonitor  
ActionID: <value>  
Channel: <value>  
[MixMonitorID:] <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - The name of the channel monitored.
- `MixMonitorID` - If a valid ID is provided, then this command will stop only that specific MixMonitor.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 ManagerAction_UserEvent

UserEvent

Synopsis

Send an arbitrary event.

Description

Send an event to manager sessions.

Syntax

```
Action: UserEvent
ActionID: <value>
UserEvent: <value>
Header1: <value>
HeaderN: <value>
```

Arguments

- **ActionID** - ActionID for this transaction. Will be returned.
- **UserEvent** - Event string to send.
- **Header1** - Content1.
- **HeaderN** - ContentN.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerAction_VoicemailRefresh

VoicemailRefresh

Synopsis

Tell Asterisk to poll mailboxes for a change

Description

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

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 AMI Events

Asterisk 12 ManagerEvent_AgentCalled

AgentCalled

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- DestChannel
- DestChannelState - A numeric code for the channel's current state, related to DestChannelStateDesc
- DestChannelStateDesc

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

See Also

- [Asterisk 12 ManagerEvent_AgentRingNoAnswer](#)
- [Asterisk 12 ManagerEvent_AgentComplete](#)
- [Asterisk 12 ManagerEvent_AgentConnect](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AgentComplete

AgentComplete

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- DestChannel

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

See Also

- [Asterisk 12 ManagerEvent_AgentCalled](#)
- [Asterisk 12 ManagerEvent_AgentConnect](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AgentConnect

AgentConnect

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- DestChannel

- DestChannelState - A numeric code for the channel's current state, related to DestChannelStateDesc
- DestChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- DestCallerIDNum
- DestCallerIDName
- DestConnectedLineNum
- DestConnectedLineName
- DestAccountCode
- DestContext
- DestExten
- DestPriority
- DestUniqueid
- Queue - The name of the queue.
- MemberName - The name of the queue member.
- Interface - The queue member's channel technology or location.
- RingTime - The time the queue member was rung, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- HoldTime - The time the channel was in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC.

See Also

- [Asterisk 12 ManagerEvent_AgentCalled](#)
- [Asterisk 12 ManagerEvent_AgentComplete](#)
- [Asterisk 12 ManagerEvent_AgentDump](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AgentDump

AgentDump

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- DestChannel
- DestChannelState - A numeric code for the channel's current state, related to DestChannelStateDesc
- DestChannelStateDesc

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

See Also

- [Asterisk 12 ManagerEvent_AgentCalled](#)
- [Asterisk 12 ManagerEvent_AgentConnect](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AgentLogin

AgentLogin

Synopsis

Raised when an Agent has logged in.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Agent - Agent ID of the agent.

See Also

- [Asterisk 12 Application_AgentLogin](#)
- [Asterisk 12 ManagerEvent_AgentLogoff](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AgentLogoff

AgentLogoff

Synopsis

Raised when an Agent has logged off.

Description

Syntax

```
Action:  
Agent: <value>  
Logintime: <value>
```

Arguments

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

See Also

- [Asterisk 12 ManagerEvent_AgentLogin](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AgentRingNoAnswer

AgentRingNoAnswer

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- DestChannel
- DestChannelState - A numeric code for the channel's current state, related to DestChannelStateDesc

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

See Also

- [Asterisk 12 ManagerEvent_AgentCalled](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Agents

Agents

Synopsis

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

Description

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

Syntax

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

Arguments

- Agent - Agent ID of the agent.
- Name - User friendly name of the agent.
- Status - Current status of the agent.
The valid values are:
 - AGENT_LOGGEDOFF
 - AGENT_IDLE
 - AGENT_ONCALL
- TalkingToChan - BRIDGEPEER value on agent channel.
Present if Status value is AGENT_ONCALL.
- CallStarted - Epoche time when the agent started talking with the caller.
Present if Status value is AGENT_ONCALL.
- LoggedInTime - Epoche time when the agent logged in.
Present if Status value is AGENT_IDLE or AGENT_ONCALL.
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName

- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- ActionID - ActionID for this transaction. Will be returned.

See Also

- [Asterisk 12 ManagerAction_Agents](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AgentsComplete

AgentsComplete

Synopsis

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

Description

Syntax

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

Arguments

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

See Also

- [Asterisk 12 ManagerAction_Agents](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AGIExecEnd

AGIExecEnd

Synopsis

Raised when a received AGI command completes processing.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Command - The AGI command as received from the external source.
- CommandId - Random identification number assigned to the execution of this command.
- ResultCode - The numeric result code from AGI
- Result - The text result reason from AGI

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AGIExecStart

AGIExecStart

Synopsis

Raised when a received AGI command starts processing.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Command - The AGI command as received from the external source.
- CommandId - Random identification number assigned to the execution of this command.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Alarm

Alarm

Synopsis

Raised when an alarm is set on a DAHDI channel.

Description

Syntax

```
Action:  
DAHDIChannel: <value>  
Alarm: <value>
```

Arguments

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



Note

This is not an Asterisk channel identifier.

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AlarmClear

AlarmClear

Synopsis

Raised when an alarm is cleared on a DAHDI channel.

Description

Syntax

```
Action:  
DAHDIChannel: <value>
```

Arguments

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



Note

This is not an Asterisk channel identifier.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AOC-D

AOC-D

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Charge
- Type
 - NotAvailable
 - Free
 - Currency

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AOC-E

AOC-E

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- ChargingAssociation
- Number
- Plan

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AOC-S

AOC-S

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Chargeable
- RateType
 - NotAvailable
 - Free

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AsyncAGIEnd

AsyncAGIEnd

Synopsis

Raised when a channel stops AsyncAGI command processing.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AsyncAGIExec

AsyncAGIExec

Synopsis

Raised when AsyncAGI completes an AGI command.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- CommandID - Optional command ID sent by the AsyncAGI server to identify the command.
- Result - URL encoded result string from the executed AGI command.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AsyncAGIStart

AsyncAGIStart

Synopsis

Raised when a channel starts AsyncAGI command processing.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Env - URL encoded string read from the AsyncAGI server.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_AttendedTransfer

AttendedTransfer

Synopsis

Raised when an attended transfer is complete.

Description

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

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

OrigBridgeUniqueid: The bridge between Alice and Bob.

SecondTransfererChannel: Alice's channel that called Voicemail.

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

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

OrigTransfererChannel: Alice's channel that called Voicemail.

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

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

SecondBridgeUniqueid: The bridge between Alice and Bob.

Syntax


```

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

```

Arguments

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



Note

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

- OrigTransfererChannel

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

- Link - The transfer was accomplished by linking two bridges together using a local channel pair.
- Threeway - The transfer was accomplished by placing all parties into a threeway call.
- Fail - The transfer failed.
- DestBridgeUniqueid - Indicates the surviving bridge when bridges were merged to complete the transfer



Note

This header is only present when *DestType* is Bridge or Threeway

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



Note

This header is only present when *DestType* is App

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

**Note**

This header is only present when *DestType* is *Threeway*

See Also**Import Version**

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_BlindTransfer

BlindTransfer

Synopsis

Raised when a blind transfer is complete.

Description

Syntax

```
Action:
Result: <value>
TransfererChannel: <value>
TransfererChannelState: <value>
TransfererChannelStateDesc: <value>
TransfererCallerIDNum: <value>
TransfererCallerIDName: <value>
TransfererConnectedLineNum: <value>
TransfererConnectedLineName: <value>
TransfererAccountCode: <value>
TransfererContext: <value>
TransfererExten: <value>
TransfererPriority: <value>
TransfererUniqueid: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
IsExternal: <value>
Context: <value>
Extension: <value>
```

Arguments

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



Note

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

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

- `TransfererContext`
- `TransfererExten`
- `TransfererPriority`
- `TransfererUniqueid`
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its `BridgeCreator` if applicable
- `BridgeNumChannels` - Number of channels in the bridge
- `IsExternal` - Indicates if the transfer was performed outside of Asterisk. For instance, a channel protocol native transfer is external. A DTMF transfer is internal.
 - Yes
 - No
- `Context` - Destination context for the blind transfer.
- `Extension` - Destination extension for the blind transfer.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_BridgeCreate

BridgeCreate

Synopsis

Raised when a bridge is created.

Description

Syntax

```
Action:  
BridgeUniqueId: <value>  
BridgeType: <value>  
BridgeTechnology: <value>  
BridgeCreator: <value>  
BridgeName: <value>  
BridgeNumChannels: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_BridgeDestroy

BridgeDestroy

Synopsis

Raised when a bridge is destroyed.

Description

Syntax

```
Action:  
BridgeUniqueId: <value>  
BridgeType: <value>  
BridgeTechnology: <value>  
BridgeCreator: <value>  
BridgeName: <value>  
BridgeNumChannels: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_BridgeEnter

BridgeEnter

Synopsis

Raised when a channel enters a bridge.

Description

Syntax

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

Arguments

- BridgeUniqueid
- BridgeType - The type of bridge
- BridgeTechnology - Technology in use by the bridge
- BridgeCreator - Entity that created the bridge if applicable
- BridgeName - Name used to refer to the bridge by its BridgeCreator if applicable
- BridgeNumChannels - Number of channels in the bridge
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- SwapUniqueid - The uniqueid of the channel being swapped out of the bridge

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_BridgeLeave

BridgeLeave

Synopsis

Raised when a channel leaves a bridge.

Description

Syntax

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

Arguments

- BridgeUniqueid
- BridgeType - The type of bridge
- BridgeTechnology - Technology in use by the bridge
- BridgeCreator - Entity that created the bridge if applicable
- BridgeName - Name used to refer to the bridge by its BridgeCreator if applicable
- BridgeNumChannels - Number of channels in the bridge
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_BridgeMerge

BridgeMerge

Synopsis

Raised when two bridges are merged.

Description

Syntax

```
Action:  
ToBridgeUniqueId: <value>  
ToBridgeType: <value>  
ToBridgeTechnology: <value>  
ToBridgeCreator: <value>  
ToBridgeName: <value>  
ToBridgeNumChannels: <value>  
FromBridgeUniqueId: <value>  
FromBridgeType: <value>  
FromBridgeTechnology: <value>  
FromBridgeCreator: <value>  
FromBridgeName: <value>  
FromBridgeNumChannels: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_ChanSpyStart

ChanSpyStart

Synopsis

Raised when one channel begins spying on another channel.

Description

Syntax

```
Action:
SpyerChannel: <value>
SpyerChannelState: <value>
SpyerChannelStateDesc: <value>
SpyerCallerIDNum: <value>
SpyerCallerIDName: <value>
SpyerConnectedLineNum: <value>
SpyerConnectedLineName: <value>
SpyerAccountCode: <value>
SpyerContext: <value>
SpyerExten: <value>
SpyerPriority: <value>
SpyerUniqueid: <value>
SpjeeChannel: <value>
SpjeeChannelState: <value>
SpjeeChannelStateDesc: <value>
SpjeeCallerIDNum: <value>
SpjeeCallerIDName: <value>
SpjeeConnectedLineNum: <value>
SpjeeConnectedLineName: <value>
SpjeeAccountCode: <value>
SpjeeContext: <value>
SpjeeExten: <value>
SpjeePriority: <value>
SpjeeUniqueid: <value>
```

Arguments

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

- OffHook
- Dialing
- Ring
- Ringing
- Up
- Busy
- Dialing Offhook
- Pre-ring
- Unknown
- SpyeeCallerIDNum
- SpyeeCallerIDName
- SpyeeConnectedLineNum
- SpyeeConnectedLineName
- SpyeeAccountCode
- SpyeeContext
- SpyeeExten
- SpyeePriority
- SpyeeUniqueid

See Also

- [Asterisk 12 Application_ChanSpyStop](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_ChanSpyStop

ChanSpyStop

Synopsis

Raised when a channel has stopped spying.

Description

Syntax

```
Action:
SpyerChannel: <value>
SpyerChannelState: <value>
SpyerChannelStateDesc: <value>
SpyerCallerIDNum: <value>
SpyerCallerIDName: <value>
SpyerConnectedLineNum: <value>
SpyerConnectedLineName: <value>
SpyerAccountCode: <value>
SpyerContext: <value>
SpyerExten: <value>
SpyerPriority: <value>
SpyerUniqueid: <value>
SpjeeChannel: <value>
SpjeeChannelState: <value>
SpjeeChannelStateDesc: <value>
SpjeeCallerIDNum: <value>
SpjeeCallerIDName: <value>
SpjeeConnectedLineNum: <value>
SpjeeConnectedLineName: <value>
SpjeeAccountCode: <value>
SpjeeContext: <value>
SpjeeExten: <value>
SpjeePriority: <value>
SpjeeUniqueid: <value>
```

Arguments

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

- OffHook
- Dialing
- Ring
- Ringing
- Up
- Busy
- Dialing Offhook
- Pre-ring
- Unknown
- SpyeeCallerIDNum
- SpyeeCallerIDName
- SpyeeConnectedLineNum
- SpyeeConnectedLineName
- SpyeeAccountCode
- SpyeeContext
- SpyeeExten
- SpyeePriority
- SpyeeUniqueid

See Also

- [Asterisk 12 Application_ChanSpyStart](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_ConfbridgeEnd

ConfbridgeEnd

Synopsis

Raised when a conference ends.

Description

Syntax

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

Arguments

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

See Also

- [Asterisk 12 ManagerEvent_ConfbridgeStart](#)
- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_ConfbridgeJoin

ConfbridgeJoin

Synopsis

Raised when a channel joins a Confbridge conference.

Description

Syntax

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

Arguments

- Conference - The name of the Confbridge conference.
- BridgeUniqueid
- BridgeType - The type of bridge
- BridgeTechnology - Technology in use by the bridge
- BridgeCreator - Entity that created the bridge if applicable
- BridgeName - Name used to refer to the bridge by its BridgeCreator if applicable
- BridgeNumChannels - Number of channels in the bridge
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

- [Asterisk 12 ManagerEvent_ConfbridgeLeave](#)
- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_ConfbridgeLeave

ConfbridgeLeave

Synopsis

Raised when a channel leaves a Confbridge conference.

Description

Syntax

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

Arguments

- Conference - The name of the Confbridge conference.
- BridgeUniqueid
- BridgeType - The type of bridge
- BridgeTechnology - Technology in use by the bridge
- BridgeCreator - Entity that created the bridge if applicable
- BridgeName - Name used to refer to the bridge by its BridgeCreator if applicable
- BridgeNumChannels - Number of channels in the bridge
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

- [Asterisk 12 ManagerEvent_ConfbridgeJoin](#)
- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_ConfbridgeMute

ConfbridgeMute

Synopsis

Raised when a Confbridge participant mutes.

Description

Syntax

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

Arguments

- Conference - The name of the Confbridge conference.
- BridgeUniqueid
- BridgeType - The type of bridge
- BridgeTechnology - Technology in use by the bridge
- BridgeCreator - Entity that created the bridge if applicable
- BridgeName - Name used to refer to the bridge by its BridgeCreator if applicable
- BridgeNumChannels - Number of channels in the bridge
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

- [Asterisk 12 ManagerEvent_ConfbridgeUnmute](#)
- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_ConfbridgeRecord

ConfbridgeRecord

Synopsis

Raised when a conference starts recording.

Description

Syntax

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

Arguments

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

See Also

- [Asterisk 12 ManagerEvent_ConfbridgeStopRecord](#)
- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_ConfbridgeStart

ConfbridgeStart

Synopsis

Raised when a conference starts.

Description

Syntax

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

Arguments

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

See Also

- [Asterisk 12 ManagerEvent_ConfbridgeEnd](#)
- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_ConfbridgeStopRecord

ConfbridgeStopRecord

Synopsis

Raised when a conference that was recording stops recording.

Description

Syntax

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

Arguments

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

See Also

- [Asterisk 12 ManagerEvent_ConfbridgeRecord](#)
- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_ConfbridgeTalking

ConfbridgeTalking

Synopsis

Raised when a confbridge participant unmutes.

Description

Syntax

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

Arguments

- Conference - The name of the Confbridge conference.
- BridgeUniqueid
- BridgeType - The type of bridge
- BridgeTechnology - Technology in use by the bridge
- BridgeCreator - Entity that created the bridge if applicable
- BridgeName - Name used to refer to the bridge by its BridgeCreator if applicable
- BridgeNumChannels - Number of channels in the bridge
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

- TalkingStatus
 - on
 - off

See Also

- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_ConfbridgeUnmute

ConfbridgeUnmute

Synopsis

Raised when a confbridge participant unmutes.

Description

Syntax

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

Arguments

- Conference - The name of the Confbridge conference.
- BridgeUniqueid
- BridgeType - The type of bridge
- BridgeTechnology - Technology in use by the bridge
- BridgeCreator - Entity that created the bridge if applicable
- BridgeName - Name used to refer to the bridge by its BridgeCreator if applicable
- BridgeNumChannels - Number of channels in the bridge
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

- [Asterisk 12 ManagerEvent_ConfbridgeMute](#)
- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 ManagerEvent_DAHDIChannel

DAHDIChannel

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- DAHDISpan - The DAHDI span associated with this channel.
- DAHDIChannel - The DAHDI channel associated with this channel.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_DialBegin

DialBegin

Synopsis

Raised when a dial action has started.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- DestChannel
- DestChannelState - A numeric code for the channel's current state, related to DestChannelStateDesc
- DestChannelStateDesc
 - Down

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

See Also

- [Asterisk 12 Application_Dial](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_DialEnd

DialEnd

Synopsis

Raised when a dial action has completed.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- DestChannel
- DestChannelState - A numeric code for the channel's current state, related to DestChannelStateDesc
- DestChannelStateDesc
 - Down

- Rsrvd
- OffHook
- Dialing
- Ring
- Ringing
- Up
- Busy
- Dialing Offhook
- Pre-ring
- Unknown
- DestCallerIDNum
- DestCallerIDName
- DestConnectedLineNum
- DestConnectedLineName
- DestAccountCode
- DestContext
- DestExten
- DestPriority
- DestUniqueid
- DialStatus - The result of the dial operation.
 - ANSWER
 - BUSY
 - CANCEL
 - CHANUNAVAIL
 - CONGESTION
 - NOANSWER

See Also

- [Asterisk 12 Application_Dial](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_DNDState

DNDState

Synopsis

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

Description

Syntax

```
Action:  
DAHDIChannel: <value>  
Status: <value>
```

Arguments

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



Note

This is not an Asterisk channel identifier.

- Status
 - enabled
 - disabled

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_DTMFBegin

DTMFBegin

Synopsis

Raised when a DTMF digit has started on a channel.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Digit - DTMF digit received or transmitted (0-9, A-E, # or *)
- Direction
 - Received
 - Sent

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_DTMFEnd

DTMFEnd

Synopsis

Raised when a DTMF digit has ended on a channel.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Digit - DTMF digit received or transmitted (0-9, A-E, # or *)
- DurationMs - Duration (in milliseconds) DTMF was sent/received
- Direction
 - Received
 - Sent

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_ExtensionStatus

ExtensionStatus

Synopsis

Raised when an extension state has changed.

Description

Syntax

```
Action:  
Exten: <value>  
Context: <value>  
Hint: <value>  
Status: <value>
```

Arguments

- Exten
- Context
- Hint
- Status

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_FAXStatus

FAXStatus

Synopsis

Raised periodically during a fax transmission.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Operation
 - gateway
 - receive
 - send
- Status - A text message describing the current status of the fax
- LocalStationID - The value of the LOCALSTATIONID channel variable
- FileName - The files being affected by the fax operation

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_FullyBooted

FullyBooted

Synopsis

Raised when all Asterisk initialization procedures have finished.

Description

Syntax

```
Action:  
Status: <value>
```

Arguments

- `Status` - Informational message

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Hangup

Hangup

Synopsis

Raised when a channel is hung up.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Cause - A numeric cause code for why the channel was hung up.
- Cause-txt - A description of why the channel was hung up.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_HangupHandlerPop

HangupHandlerPop

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Handler - Hangup handler parameter string passed to the Gosub application.

See Also

- [Asterisk 12 ManagerEvent_HangupHandlerPush](#)
- [Asterisk 12 Function_CHANNEL](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_HangupHandlerPush

HangupHandlerPush

Synopsis

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

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Handler - Hangup handler parameter string passed to the Gosub application.

See Also

- [Asterisk 12 ManagerEvent_HangupHandlerPop](#)
- [Asterisk 12 Function_CHANNEL](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_HangupHandlerRun

HangupHandlerRun

Synopsis

Raised when a hangup handler is about to be called.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Handler - Hangup handler parameter string passed to the Gosub application.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_HangupRequest

HangupRequest

Synopsis

Raised when a hangup is requested.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Cause - A numeric cause code for why the channel was hung up.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Hold

Hold

Synopsis

Raised when a channel goes on hold.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- MusicClass - The suggested MusicClass, if provided.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_LocalBridge

LocalBridge

Synopsis

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

Description

Syntax

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

Arguments

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

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_LocalOptimizationBegin

LocalOptimizationBegin

Synopsis

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

Description

Syntax

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

Arguments

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

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

See Also

- [Asterisk 12 ManagerEvent_LocalOptimizationEnd](#)
- [Asterisk 12 ManagerAction_LocalOptimizeAway](#)

Import Version

Asterisk 12 ManagerEvent_LocalOptimizationEnd

LocalOptimizationEnd

Synopsis

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

Description

Syntax

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

Arguments

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

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

See Also

- [Asterisk 12 ManagerEvent_LocalOptimizationBegin](#)
- [Asterisk 12 ManagerAction_LocalOptimizeAway](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_LogChannel

LogChannel

Synopsis

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

Description

Syntax

```
Action:  
Channel: <value>  
Enabled: <value>
```

Arguments

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

See Also

Synopsis

Raised when a logging channel is disabled.

Description

Syntax

```
Action:  
Channel: <value>  
Enabled: <value>  
Reason: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MCID

MCID

Synopsis

Published when a malicious call ID request arrives.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName

- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- MCallerIDNumValid
- MCallerIDNum
- MCallerIDton
- MCallerIDNumPlan
- MCallerIDNumPres
- MCallerIDNameValid
- MCallerIDName
- MCallerIDNameCharSet
- MCallerIDNamePres
- MCallerIDSubaddr
- MCallerIDSubaddrType
- MCallerIDSubaddrOdd
- MCallerIDPres
- MConnectedIDNumValid
- MConnectedIDNum
- MConnectedIDton
- MConnectedIDNumPlan
- MConnectedIDNumPres
- MConnectedIDNameValid
- MConnectedIDName
- MConnectedIDNameCharSet
- MConnectedIDNamePres
- MConnectedIDSubaddr
- MConnectedIDSubaddrType
- MConnectedIDSubaddrOdd
- MConnectedIDPres

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MeetmeEnd

MeetmeEnd

Synopsis

Raised when a MeetMe conference ends.

Description

Syntax

```
Action:  
Meetme: <value>
```

Arguments

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

See Also

- [Asterisk 12 ManagerEvent_MeetmeJoin](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MeetmeJoin

MeetmeJoin

Synopsis

Raised when a user joins a MeetMe conference.

Description

Syntax

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

Arguments

- **Meetme** - The identifier for the MeetMe conference.
- **Usernum** - The identifier of the MeetMe user who joined.
- **Channel**
- **ChannelState** - A numeric code for the channel's current state, related to ChannelStateDesc
- **ChannelStateDesc**
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- **CallerIDNum**
- **CallerIDName**
- **ConnectedLineNum**
- **ConnectedLineName**
- **AccountCode**
- **Context**
- **Exten**
- **Priority**
- **Uniqueid**

See Also

- [Asterisk 12 ManagerEvent_MeetmeLeave](#)
- [Asterisk 12 Application_MeetMe](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MeetmeLeave

MeetmeLeave

Synopsis

Raised when a user leaves a MeetMe conference.

Description

Syntax

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

Arguments

- **Meetme** - The identifier for the MeetMe conference.
- **Usernum** - The identifier of the MeetMe user who joined.
- **Channel**
- **ChannelState** - A numeric code for the channel's current state, related to ChannelStateDesc
- **ChannelStateDesc**
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- **CallerIDNum**
- **CallerIDName**
- **ConnectedLineNum**
- **ConnectedLineName**
- **AccountCode**
- **Context**
- **Exten**
- **Priority**
- **Uniqueid**
- **Duration** - The length of time in seconds that the Meetme user was in the conference.

See Also

- [Asterisk 12 ManagerEvent_MeetmeJoin](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MeetmeMute

MeetmeMute

Synopsis

Raised when a MeetMe user is muted or unmuted.

Description

Syntax

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

Arguments

- Meetme - The identifier for the MeetMe conference.
- Usernum - The identifier of the MeetMe user who joined.
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Duration - The length of time in seconds that the Meetme user has been in the conference at the time of this event.
- Status
 - on
 - off

See Also

Import Version

Asterisk 12 ManagerEvent_MeetmeTalking

MeetmeTalking

Synopsis

Raised when a MeetMe user begins or ends talking.

Description

Syntax

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

Arguments

- Meetme - The identifier for the MeetMe conference.
- Usernum - The identifier of the MeetMe user who joined.
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Duration - The length of time in seconds that the Meetme user has been in the conference at the time of this event.
- Status
 - on
 - off

See Also

Import Version

Asterisk 12 ManagerEvent_MeetmeTalkRequest

MeetmeTalkRequest

Synopsis

Raised when a MeetMe user has started talking.

Description

Syntax

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

Arguments

- Meetme - The identifier for the MeetMe conference.
- Usernum - The identifier of the MeetMe user who joined.
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Duration - The length of time in seconds that the Meetme user has been in the conference at the time of this event.
- Status
 - on
 - off

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MessageWaiting

MessageWaiting

Synopsis

Raised when the state of messages in a voicemail mailbox has changed or when a channel has finished interacting with a mailbox.

Description



Note

The Channel related parameters are only present if a channel was involved in the manipulation of a mailbox. If no channel is involved, the parameters are not included with the event.

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Mailbox: <value>
Waiting: <value>
New: <value>
Old: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Mailbox - The mailbox with the new message, specified as mailbox@context
- Waiting - Whether or not the mailbox has messages waiting for it.
- New - The number of new messages.
- Old - The number of old messages.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MiniVoiceMail

MiniVoiceMail

Synopsis

Raised when a notification is sent out by a MiniVoiceMail application

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Action: <value>
Mailbox: <value>
Counter: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Action - What action was taken. Currently, this will always be SentNotification
- Mailbox - The mailbox that the notification was about, specified as mailbox@context
- Counter - A message counter derived from the MVM_COUNTER channel variable.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MonitorStart

MonitorStart

Synopsis

Raised when monitoring has started on a channel.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

- [Asterisk 12 ManagerEvent_MonitorStop](#)
- [Asterisk 12 Application_Monitor](#)
- [Asterisk 12 ManagerAction_Monitor](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MonitorStop

MonitorStop

Synopsis

Raised when monitoring has stopped on a channel.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

- [Asterisk 12 ManagerEvent_MonitorStart](#)
- [Asterisk 12 Application_StopMonitor](#)
- [Asterisk 12 ManagerAction_StopMonitor](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MusicOnHoldStart

MusicOnHoldStart

Synopsis

Raised when music on hold has started on a channel.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Class - The class of music being played on the channel

See Also

- [Asterisk 12 ManagerEvent_MusicOnHoldStop](#)
- [Asterisk 12 Application_MusicOnHold](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_MusicOnHoldStop

MusicOnHoldStop

Synopsis

Raised when music on hold has stopped on a channel.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

- [Asterisk 12 ManagerEvent_MusicOnHoldStart](#)
- [Asterisk 12 Application_StopMusicOnHold](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_NewAccountCode

NewAccountCode

Synopsis

Raised when a Channel's AccountCode is changed.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- OldAccountCode - The channel's previous account code

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_NewCallerid

NewCallerid

Synopsis

Raised when a channel receives new Caller ID information.

Description

Syntax

```
Action:  
Channel: <value>  
ChannelState: <value>  
ChannelStateDesc: <value>  
CallerIDNum: <value>  
CallerIDName: <value>  
ConnectedLineNum: <value>  
ConnectedLineName: <value>  
AccountCode: <value>  
Context: <value>  
Exten: <value>  
Priority: <value>  
Uniqueid: <value>  
CID-CallingPres: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- CID-CallingPres - A description of the Caller ID presentation.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Newchannel

Newchannel

Synopsis

Raised when a new channel is created.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_NewExten

NewExten

Synopsis

Raised when a channel enters a new context, extension, priority.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Extension: <value>
Application: <value>
AppData: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Extension - Deprecated in 12, but kept for backward compatability. Please use 'Exten' instead.
- Application - The application about to be executed.
- AppData - The data to be passed to the application.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Newstate

Newstate

Synopsis

Raised when a channel's state changes.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_OriginateResponse

OriginateResponse

Synopsis

Raised in response to an Originate command.

Description

Syntax

```
Action:
[ActionID:] <value>
Resonse: <value>
Channel: <value>
Context: <value>
Exten: <value>
Reason: <value>
Uniqueid: <value>
CallerIDNum: <value>
CallerIDName: <value>
```

Arguments

- ActionID
- Resonse
 - Failure
 - Success
- Channel
- Context
- Exten
- Reason
- Uniqueid
- CallerIDNum
- CallerIDName

See Also

- [Asterisk 12 ManagerAction_Originate](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_ParkedCall

ParkedCall

Synopsis

Raised when a channel is parked.

Description

Syntax

```
Action:
ParkeeChannel: <value>
ParkeeChannelState: <value>
ParkeeChannelStateDesc: <value>
ParkeeCallerIDNum: <value>
ParkeeCallerIDName: <value>
ParkeeConnectedLineNum: <value>
ParkeeConnectedLineName: <value>
ParkeeAccountCode: <value>
ParkeeContext: <value>
ParkeeExten: <value>
ParkeePriority: <value>
ParkeeUniqueid: <value>
ParkerDialString: <value>
Parkinglot: <value>
ParkingSpace: <value>
ParkingTimeout: <value>
ParkingDuration: <value>
```

Arguments

- ParkeeChannel
- ParkeeChannelState - A numeric code for the channel's current state, related to ParkeeChannelStateDesc
- ParkeeChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- ParkeeCallerIDNum
- ParkeeCallerIDName
- ParkeeConnectedLineNum
- ParkeeConnectedLineName
- ParkeeAccountCode
- ParkeeContext
- ParkeeExten
- ParkeePriority
- ParkeeUniqueid
- ParkerDialString - Dial String that can be used to call back the parker on ParkingTimeout.
- Parkinglot - Name of the parking lot that the parkee is parked in
- ParkingSpace - Parking Space that the parkee is parked in
- ParkingTimeout - Time remaining until the parkee is forcefully removed from parking in seconds
- ParkingDuration - Time the parkee has been in the parking bridge (in seconds)

See Also

Import Version

Asterisk 12 ManagerEvent_ParkedCallGiveUp

ParkedCallGiveUp

Synopsis

Raised when a channel leaves a parking lot because it hung up without being answered.

Description

Syntax

```
Action:
ParkeeChannel: <value>
ParkeeChannelState: <value>
ParkeeChannelStateDesc: <value>
ParkeeCallerIDNum: <value>
ParkeeCallerIDName: <value>
ParkeeConnectedLineNum: <value>
ParkeeConnectedLineName: <value>
ParkeeAccountCode: <value>
ParkeeContext: <value>
ParkeeExten: <value>
ParkeePriority: <value>
ParkeeUniqueid: <value>
ParkerChannel: <value>
ParkerChannelState: <value>
ParkerChannelStateDesc: <value>
ParkerCallerIDNum: <value>
ParkerCallerIDName: <value>
ParkerConnectedLineNum: <value>
ParkerConnectedLineName: <value>
ParkerAccountCode: <value>
ParkerContext: <value>
ParkerExten: <value>
ParkerPriority: <value>
ParkerUniqueid: <value>
ParkerDialString: <value>
Parkinglot: <value>
ParkingSpace: <value>
ParkingTimeout: <value>
ParkingDuration: <value>
```

Arguments

- ParkeeChannel
- ParkeeChannelState - A numeric code for the channel's current state, related to ParkeeChannelStateDesc
- ParkeeChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- ParkeeCallerIDNum
- ParkeeCallerIDName
- ParkeeConnectedLineNum
- ParkeeConnectedLineName
- ParkeeAccountCode
- ParkeeContext
- ParkeeExten
- ParkeePriority
- ParkeeUniqueid
- ParkerChannel

- `ParkerChannelState` - A numeric code for the channel's current state, related to `ParkerChannelStateDesc`
- `ParkerChannelStateDesc`
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- `ParkerCallerIDNum`
- `ParkerCallerIDName`
- `ParkerConnectedLineNum`
- `ParkerConnectedLineName`
- `ParkerAccountCode`
- `ParkerContext`
- `ParkerExten`
- `ParkerPriority`
- `ParkerUniqueid`
- `ParkerDialString` - Dial String that can be used to call back the parker on `ParkingTimeout`.
- `Parkinglot` - Name of the parking lot that the parkee is parked in
- `ParkingSpace` - Parking Space that the parkee is parked in
- `ParkingTimeout` - Time remaining until the parkee is forcefully removed from parking in seconds
- `ParkingDuration` - Time the parkee has been in the parking bridge (in seconds)

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_ParkedCallTimeOut

ParkedCallTimeOut

Synopsis

Raised when a channel leaves a parking lot due to reaching the time limit of being parked.

Description

Syntax

```
Action:
ParkeeChannel: <value>
ParkeeChannelState: <value>
ParkeeChannelStateDesc: <value>
ParkeeCallerIDNum: <value>
ParkeeCallerIDName: <value>
ParkeeConnectedLineNum: <value>
ParkeeConnectedLineName: <value>
ParkeeAccountCode: <value>
ParkeeContext: <value>
ParkeeExten: <value>
ParkeePriority: <value>
ParkeeUniqueid: <value>
ParkerChannel: <value>
ParkerChannelState: <value>
ParkerChannelStateDesc: <value>
ParkerCallerIDNum: <value>
ParkerCallerIDName: <value>
ParkerConnectedLineNum: <value>
ParkerConnectedLineName: <value>
ParkerAccountCode: <value>
ParkerContext: <value>
ParkerExten: <value>
ParkerPriority: <value>
ParkerUniqueid: <value>
ParkerDialString: <value>
Parkinglot: <value>
ParkingSpace: <value>
ParkingTimeout: <value>
ParkingDuration: <value>
```

Arguments

- ParkeeChannel
- ParkeeChannelState - A numeric code for the channel's current state, related to ParkeeChannelStateDesc
- ParkeeChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- ParkeeCallerIDNum
- ParkeeCallerIDName
- ParkeeConnectedLineNum
- ParkeeConnectedLineName
- ParkeeAccountCode
- ParkeeContext
- ParkeeExten
- ParkeePriority
- ParkeeUniqueid
- ParkerChannel

- `ParkerChannelState` - A numeric code for the channel's current state, related to `ParkerChannelStateDesc`
- `ParkerChannelStateDesc`
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- `ParkerCallerIDNum`
- `ParkerCallerIDName`
- `ParkerConnectedLineNum`
- `ParkerConnectedLineName`
- `ParkerAccountCode`
- `ParkerContext`
- `ParkerExten`
- `ParkerPriority`
- `ParkerUniqueid`
- `ParkerDialString` - Dial String that can be used to call back the parker on `ParkingTimeout`.
- `Parkinglot` - Name of the parking lot that the parkee is parked in
- `ParkingSpace` - Parking Space that the parkee is parked in
- `ParkingTimeout` - Time remaining until the parkee is forcefully removed from parking in seconds
- `ParkingDuration` - Time the parkee has been in the parking bridge (in seconds)

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_PeerStatus

PeerStatus

Synopsis

Raised when the state of a peer changes.

Description

Syntax

```
Action:  
ChannelType: <value>  
Peer: <value>  
PeerStatus: <value>  
Cause: <value>  
Address: <value>  
Port: <value>  
Time: <value>
```

Arguments

- **ChannelType** - The channel technology of the peer.
- **Peer** - The name of the peer (including channel technology).
- **PeerStatus** - New status of the peer.
 - Unknown
 - Registered
 - Unregistered
 - Rejected
 - Lagged
- **Cause** - The reason the status has changed.
- **Address** - New address of the peer.
- **Port** - New port for the peer.
- **Time** - Time it takes to reach the peer and receive a response.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Pickup

Pickup

Synopsis

Raised when a call pickup occurs.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
TargetChannel: <value>
TargetChannelState: <value>
TargetChannelStateDesc: <value>
TargetCallerIDNum: <value>
TargetCallerIDName: <value>
TargetConnectedLineNum: <value>
TargetConnectedLineName: <value>
TargetAccountCode: <value>
TargetContext: <value>
TargetExten: <value>
TargetPriority: <value>
TargetUniqueid: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- TargetChannel
- TargetChannelState - A numeric code for the channel's current state, related to TargetChannelStateDesc
- TargetChannelStateDesc
 - Down
 - Rsrvd

- OffHook
- Dialing
- Ring
- Ringing
- Up
- Busy
- Dialing Offhook
- Pre-ring
- Unknown
- TargetCallerIDNum
- TargetCallerIDName
- TargetConnectedLineNum
- TargetConnectedLineName
- TargetAccountCode
- TargetContext
- TargetExten
- TargetPriority
- TargetUniqueid

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_PresenceStatus

PresenceStatus

Synopsis

Raised when a presence state has changed.

Description

Syntax

```
Action:  
Exten: <value>  
Context: <value>  
Hint: <value>  
Status: <value>  
Subtype: <value>  
Message: <value>
```

Arguments

- Exten
- Context
- Hint
- Status
- Subtype
- Message

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404320

Asterisk 12 ManagerEvent_QueueCallerAbandon

QueueCallerAbandon

Synopsis

Raised when a caller abandons the queue.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Queue: <value>
Position: <value>
OriginalPosition: <value>
HoldTime: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Queue - The name of the queue.
- Position - This channel's current position in the queue.
- OriginalPosition - The channel's original position in the queue.
- HoldTime - The time the channel was in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_QueueCallerJoin

QueueCallerJoin

Synopsis

Raised when a caller joins a Queue.

Description

Syntax

```
Action:  
Channel: <value>  
ChannelState: <value>  
ChannelStateDesc: <value>  
CallerIDNum: <value>  
CallerIDName: <value>  
ConnectedLineNum: <value>  
ConnectedLineName: <value>  
AccountCode: <value>  
Context: <value>  
Exten: <value>  
Priority: <value>  
Uniqueid: <value>  
Queue: <value>  
Position: <value>  
Count: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Queue - The name of the queue.
- Position - This channel's current position in the queue.
- Count - The total number of channels in the queue.

See Also

- [Asterisk 12 ManagerEvent_QueueCallerLeave](#)
- [Asterisk 12 Application_Queue](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_QueueCallerLeave

QueueCallerLeave

Synopsis

Raised when a caller leaves a Queue.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Queue: <value>
Count: <value>
Position: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Queue - The name of the queue.
- Count - The total number of channels in the queue.
- Position - This channel's current position in the queue.

See Also

- [Asterisk 12 ManagerEvent_QueueCallerJoin](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_QueueMemberAdded

QueueMemberAdded

Synopsis

Raised when a member is added to the queue.

Description

Syntax

```
Action:  
Queue: <value>  
MemberName: <value>  
Interface: <value>  
StateInterface: <value>  
Membership: <value>  
Penalty: <value>  
CallsTaken: <value>  
LastCall: <value>  
Status: <value>  
Paused: <value>  
Ringinuse: <value>
```

Arguments

- Queue - The name of the queue.
- MemberName - The name of the queue member.
- Interface - The queue member's channel technology or location.
- StateInterface - Channel technology or location from which to read device state changes.
- Membership
 - dynamic
 - realtime
 - static
- Penalty - The penalty associated with the queue member.
- CallsTaken - The number of calls this queue member has serviced.
- LastCall - The time this member last took a call, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- Status - The numeric device state status of the queue member.
 - 0 - AST_DEVICE_UNKNOWN
 - 1 - AST_DEVICE_NOT_INUSE
 - 2 - AST_DEVICE_INUSE
 - 3 - AST_DEVICE_BUSY
 - 4 - AST_DEVICE_INVALID
 - 5 - AST_DEVICE_UNAVAILABLE
 - 6 - AST_DEVICE_RINGING
 - 7 - AST_DEVICE_RINGINUSE
 - 8 - AST_DEVICE_ONHOLD
- Paused
 - 0
 - 1
- Ringinuse
 - 0
 - 1

See Also

- [Asterisk 12 ManagerEvent_QueueMemberRemoved](#)
- [Asterisk 12 Application_AddQueueMember](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_QueueMemberPaused

QueueMemberPaused

Synopsis

Raised when a member is paused/unpaused in the queue.

Description

Syntax

```
Action:
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
Reason: <value>
```

Arguments

- Queue - The name of the queue.
- MemberName - The name of the queue member.
- Interface - The queue member's channel technology or location.
- StateInterface - Channel technology or location from which to read device state changes.
- Membership
 - dynamic
 - realtime
 - static
- Penalty - The penalty associated with the queue member.
- CallsTaken - The number of calls this queue member has serviced.
- LastCall - The time this member last took a call, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- Status - The numeric device state status of the queue member.
 - 0 - AST_DEVICE_UNKNOWN
 - 1 - AST_DEVICE_NOT_INUSE
 - 2 - AST_DEVICE_INUSE
 - 3 - AST_DEVICE_BUSY
 - 4 - AST_DEVICE_INVALID
 - 5 - AST_DEVICE_UNAVAILABLE
 - 6 - AST_DEVICE_RINGING
 - 7 - AST_DEVICE_RINGINUSE
 - 8 - AST_DEVICE_ONHOLD
- Paused
 - 0
 - 1
- Ringinuse
 - 0
 - 1
- Reason - The reason a member was paused.

See Also

- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnPauseQueueMember](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_QueueMemberPenalty

QueueMemberPenalty

Synopsis

Raised when a member's penalty is changed.

Description

Syntax

```
Action:
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
```

Arguments

- Queue - The name of the queue.
- MemberName - The name of the queue member.
- Interface - The queue member's channel technology or location.
- StateInterface - Channel technology or location from which to read device state changes.
- Membership
 - dynamic
 - realtime
 - static
- Penalty - The penalty associated with the queue member.
- CallsTaken - The number of calls this queue member has serviced.
- LastCall - The time this member last took a call, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- Status - The numeric device state status of the queue member.
 - 0 - AST_DEVICE_UNKNOWN
 - 1 - AST_DEVICE_NOT_INUSE
 - 2 - AST_DEVICE_INUSE
 - 3 - AST_DEVICE_BUSY
 - 4 - AST_DEVICE_INVALID
 - 5 - AST_DEVICE_UNAVAILABLE
 - 6 - AST_DEVICE_RINGING
 - 7 - AST_DEVICE_RINGINUSE
 - 8 - AST_DEVICE_ONHOLD
- Paused
 - 0
 - 1
- Ringinuse
 - 0
 - 1

See Also

- [Asterisk 12 Function_QUEUE_MEMBER](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_QueueMemberRemoved

QueueMemberRemoved

Synopsis

Raised when a member is removed from the queue.

Description

Syntax

```
Action:
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
```

Arguments

- Queue - The name of the queue.
- MemberName - The name of the queue member.
- Interface - The queue member's channel technology or location.
- StateInterface - Channel technology or location from which to read device state changes.
- Membership
 - dynamic
 - realtime
 - static
- Penalty - The penalty associated with the queue member.
- CallsTaken - The number of calls this queue member has serviced.
- LastCall - The time this member last took a call, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- Status - The numeric device state status of the queue member.
 - 0 - AST_DEVICE_UNKNOWN
 - 1 - AST_DEVICE_NOT_INUSE
 - 2 - AST_DEVICE_INUSE
 - 3 - AST_DEVICE_BUSY
 - 4 - AST_DEVICE_INVALID
 - 5 - AST_DEVICE_UNAVAILABLE
 - 6 - AST_DEVICE_RINGING
 - 7 - AST_DEVICE_RINGINUSE
 - 8 - AST_DEVICE_ONHOLD
- Paused
 - 0
 - 1
- Ringinuse
 - 0
 - 1

See Also

- [Asterisk 12 ManagerEvent_QueueMemberAdded](#)
- [Asterisk 12 Application_RemoveQueueMember](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_QueueMemberRinginuse

QueueMemberRinginuse

Synopsis

Raised when a member's ringinuse setting is changed.

Description

Syntax

```
Action:
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
```

Arguments

- Queue - The name of the queue.
- MemberName - The name of the queue member.
- Interface - The queue member's channel technology or location.
- StateInterface - Channel technology or location from which to read device state changes.
- Membership
 - dynamic
 - realtime
 - static
- Penalty - The penalty associated with the queue member.
- CallsTaken - The number of calls this queue member has serviced.
- LastCall - The time this member last took a call, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- Status - The numeric device state status of the queue member.
 - 0 - AST_DEVICE_UNKNOWN
 - 1 - AST_DEVICE_NOT_INUSE
 - 2 - AST_DEVICE_INUSE
 - 3 - AST_DEVICE_BUSY
 - 4 - AST_DEVICE_INVALID
 - 5 - AST_DEVICE_UNAVAILABLE
 - 6 - AST_DEVICE_RINGING
 - 7 - AST_DEVICE_RINGINUSE
 - 8 - AST_DEVICE_ONHOLD
- Paused
 - 0
 - 1
- Ringinuse
 - 0
 - 1

See Also

- [Asterisk 12 Function_QUEUE_MEMBER](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_QueueMemberStatus

QueueMemberStatus

Synopsis

Raised when a Queue member's status has changed.

Description

Syntax

```
Action:
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
```

Arguments

- Queue - The name of the queue.
- MemberName - The name of the queue member.
- Interface - The queue member's channel technology or location.
- StateInterface - Channel technology or location from which to read device state changes.
- Membership
 - dynamic
 - realtime
 - static
- Penalty - The penalty associated with the queue member.
- CallsTaken - The number of calls this queue member has serviced.
- LastCall - The time this member last took a call, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- Status - The numeric device state status of the queue member.
 - 0 - AST_DEVICE_UNKNOWN
 - 1 - AST_DEVICE_NOT_INUSE
 - 2 - AST_DEVICE_INUSE
 - 3 - AST_DEVICE_BUSY
 - 4 - AST_DEVICE_INVALID
 - 5 - AST_DEVICE_UNAVAILABLE
 - 6 - AST_DEVICE_RINGING
 - 7 - AST_DEVICE_RINGINUSE
 - 8 - AST_DEVICE_ONHOLD
- Paused
 - 0
 - 1
- Ringinuse
 - 0
 - 1

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_ReceiveFAX

ReceiveFAX

Synopsis

Raised when a receive fax operation has completed.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
LocalStationID: <value>
RemoteStationID: <value>
PagesTransferred: <value>
Resolution: <value>
TransferRate: <value>
FileName: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- LocalStationID - The value of the LOCALSTATIONID channel variable
- RemoteStationID - The value of the REMOTESTATIONID channel variable
- PagesTransferred - The number of pages that have been transferred
- Resolution - The negotiated resolution
- TransferRate - The negotiated transfer rate
- FileName - The files being affected by the fax operation

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Registry

Registry

Synopsis

Raised when an outbound registration completes.

Description

Syntax

```
Action:  
ChannelType: <value>  
Username: <value>  
Domain: <value>  
Status: <value>  
Cause: <value>
```

Arguments

- ChannelType - The type of channel that was registered (or not).
- Username - The username portion of the registration.
- Domain - The address portion of the registration.
- Status - The status of the registration request.
 - Registered
 - Unregistered
 - Rejected
 - Failed
- Cause - What caused the rejection of the request, if available.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Reload

Reload

Synopsis

Raised when a module has been reloaded in Asterisk.

Description

Syntax

```
Action:  
Module: <value>  
Status: <value>
```

Arguments

- **Module** - The name of the module that was reloaded, or `All` if all modules were reloaded
- **Status** - The numeric status code denoting the success or failure of the reload request.
 - 0 - Success
 - 1 - Request queued
 - 2 - Module not found
 - 3 - Error
 - 4 - Reload already in progress
 - 5 - Module uninitialized
 - 6 - Reload not supported

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Rename

Rename

Synopsis

Raised when the name of a channel is changed.

Description

Syntax

```
Action:  
Channel: <value>  
Newname: <value>  
Uniqueid: <value>
```

Arguments

- Channel
- Newname
- Uniqueid

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_RTCPReceived

RTCPReceived

Synopsis

Raised when an RTCP packet is received.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
SSRC: <value>
PT: <value>
From: <value>
RTT: <value>
ReportCount: <value>
[SentNTP:] <value>
[SentRTP:] <value>
[SentPackets:] <value>
[SentOctets:] <value>
ReportXSourceSSRC: <value>
ReportXFractionLost: <value>
ReportXCumulativeLost: <value>
ReportXHighestSequence: <value>
ReportXSequenceNumberCycles: <value>
ReportXIAJitter: <value>
ReportXLSR: <value>
ReportXDLSR: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- SSRC - The SSRC identifier for the remote system

- **PT** - The type of packet for this RTCP report.
 - 200 (SR)
 - 201 (RR)
- **From** - The address the report was received from.
- **RTT** - Calculated Round-Trip Time in seconds
- **ReportCount** - The number of reports that were received.
The report count determines the number of ReportX headers in the message. The X for each set of report headers will range from 0 to ReportCount - 1.
- **SentNTP** - The time the sender generated the report. Only valid when PT is 200 (SR).
- **SentRTP** - The sender's last RTP timestamp. Only valid when PT is 200 (SR).
- **SentPackets** - The number of packets the sender has sent. Only valid when PT is 200 (SR).
- **SentOctets** - The number of bytes the sender has sent. Only valid when PT is 200 (SR).
- **ReportXSourceSSRC** - The SSRC for the source of this report block.
- **ReportXFractionLost** - The fraction of RTP data packets from ReportXSourceSSRC lost since the previous SR or RR report was sent.
- **ReportXCumulativeLost** - The total number of RTP data packets from ReportXSourceSSRC lost since the beginning of reception.
- **ReportXHighestSequence** - The highest sequence number received in an RTP data packet from ReportXSourceSSRC.
- **ReportXSequenceNumberCycles** - The number of sequence number cycles seen for the RTP data received from ReportXSourceSSRC.
- **ReportXIAJitter** - An estimate of the statistical variance of the RTP data packet interarrival time, measured in timestamp units.
- **ReportXLSR** - The last SR timestamp received from ReportXSourceSSRC. If no SR has been received from ReportXSourceSSRC, then 0.
- **ReportXDLSR** - The delay, expressed in units of 1/65536 seconds, between receiving the last SR packet from ReportXSourceSSRC and sending this report.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_RTCPSent

RTCPSent

Synopsis

Raised when an RTCP packet is sent.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
SSRC: <value>
PT: <value>
To: <value>
ReportCount: <value>
[SentNTP:] <value>
[SentRTP:] <value>
[SentPackets:] <value>
[SentOctets:] <value>
ReportXSourceSSRC: <value>
ReportXFractionLost: <value>
ReportXCumulativeLost: <value>
ReportXHighestSequence: <value>
ReportXSequenceNumberCycles: <value>
ReportXIAJitter: <value>
ReportXLSR: <value>
ReportXDLSR: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- SSRC - The SSRC identifier for our stream
- PT - The type of packet for this RTCP report.

- 200(SR)
- 201(RR)
- To - The address the report is sent to.
- ReportCount - The number of reports that were sent.
The report count determines the number of ReportX headers in the message. The X for each set of report headers will range from 0 to ReportCount - 1.
- SentNTP - The time the sender generated the report. Only valid when PT is 200(SR).
- SentRTP - The sender's last RTP timestamp. Only valid when PT is 200(SR).
- SentPackets - The number of packets the sender has sent. Only valid when PT is 200(SR).
- SentOctets - The number of bytes the sender has sent. Only valid when PT is 200(SR).
- ReportXSourceSSRC - The SSRC for the source of this report block.
- ReportXFractionLost - The fraction of RTP data packets from ReportXSourceSSRC lost since the previous SR or RR report was sent.
- ReportXCumulativeLost - The total number of RTP data packets from ReportXSourceSSRC lost since the beginning of reception.
- ReportXHighestSequence - The highest sequence number received in an RTP data packet from ReportXSourceSSRC.
- ReportXSequenceNumberCycles - The number of sequence number cycles seen for the RTP data received from ReportXSourceSSRC.
- ReportXIAJitter - An estimate of the statistical variance of the RTP data packet interarrival time, measured in timestamp units.
- ReportXLSR - The last SR timestamp received from ReportXSourceSSRC. If no SR has been received from ReportXSourceSSRC, then 0.
- ReportXDLSR - The delay, expressed in units of 1/65536 seconds, between receiving the last SR packet from ReportXSourceSSRC and sending this report.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_SendFAX

SendFAX

Synopsis

Raised when a send fax operation has completed.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
LocalStationID: <value>
RemoteStationID: <value>
PagesTransferred: <value>
Resolution: <value>
TransferRate: <value>
FileName: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- LocalStationID - The value of the LOCALSTATIONID channel variable
- RemoteStationID - The value of the REMOTESTATIONID channel variable
- PagesTransferred - The number of pages that have been transferred
- Resolution - The negotiated resolution
- TransferRate - The negotiated transfer rate
- FileName - The files being affected by the fax operation

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_SessionTimeout

SessionTimeout

Synopsis

Raised when a SIP session times out.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Source - The source of the session timeout.
 - RTPTimeout
 - SIPSessionTimer

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Shutdown

Shutdown

Synopsis

Raised when Asterisk is shutdown or restarted.

Description

Syntax

```
Action:  
Shutdown: <value>  
Restart: <value>
```

Arguments

- **Shutdown** - Whether the shutdown is proceeding cleanly (all channels were hungup successfully) or uncleanly (channels will be terminated)
 - Uncleanly
 - Cleanly
- **Restart** - Whether or not a restart will occur.
 - True
 - False

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_SIPQualifyPeerDone

SIPQualifyPeerDone

Synopsis

Raised when SIPQualifyPeer has finished qualifying the specified peer.

Description

Syntax

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

Arguments

- `Peer` - The name of the peer.
- `ActionID` - This is only included if an ActionID Header was sent with the action request, in which case it will be that ActionID.

See Also

- [Asterisk 12 ManagerAction_SIPqualifypeer](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_SoftHangupRequest

SoftHangupRequest

Synopsis

Raised when a soft hangup is requested with a specific cause code.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Cause - A numeric cause code for why the channel was hung up.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_SpanAlarm

SpanAlarm

Synopsis

Raised when an alarm is set on a DAHDI span.

Description

Syntax

```
Action:  
Span: <value>  
Alarm: <value>
```

Arguments

- `Span` - The span on which the alarm occurred.
- `Alarm` - A textual description of the alarm that occurred.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_SpanAlarmClear

SpanAlarmClear

Synopsis

Raised when an alarm is cleared on a DAHDI span.

Description

Syntax

```
Action:  
Span: <value>
```

Arguments

- `Span` - The span on which the alarm was cleared.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Status

Status

Synopsis

Raised in response to a Status command.

Description

Syntax

```
Action:
[ActionID:] <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Type: <value>
DNID: <value>
TimeToHangup: <value>
BridgeID: <value>
Linkedid: <value>
Application: <value>
Data: <value>
Nativeformats: <value>
Readformat: <value>
Readtrans: <value>
Writeformat: <value>
Writetrans: <value>
Callgroup: <value>
Pickupgroup: <value>
Seconds: <value>
```

Arguments

- ActionID
- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Type - Type of channel

- **DNID** - Dialed number identifier
- **TimeToHangup** - Absolute lifetime of the channel
- **BridgeID** - Identifier of the bridge the channel is in, may be empty if not in one
- **Linkedid**
- **Application** - Application currently executing on the channel
- **Data** - Data given to the currently executing channel
- **Nativeformats** - Media formats the connected party is willing to send or receive
- **Readformat** - Media formats that frames from the channel are received in
- **Readtrans** - Translation path for media received in native formats
- **Writeformat** - Media formats that frames to the channel are accepted in
- **Writetrans** - Translation path for media sent to the connected party
- **Callgroup** - Configured call group on the channel
- **Pickupgroup** - Configured pickup group on the channel
- **Seconds** - Number of seconds the channel has been active

See Also

- [Asterisk 12 ManagerAction_Status](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_Unhold

Unhold

Synopsis

Raised when a channel goes off hold.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_UnParkedCall

UnParkedCall

Synopsis

Raised when a channel leaves a parking lot because it was retrieved from the parking lot and reconnected.

Description

Syntax

```
Action:
ParkeeChannel: <value>
ParkeeChannelState: <value>
ParkeeChannelStateDesc: <value>
ParkeeCallerIDNum: <value>
ParkeeCallerIDName: <value>
ParkeeConnectedLineNum: <value>
ParkeeConnectedLineName: <value>
ParkeeAccountCode: <value>
ParkeeContext: <value>
ParkeeExten: <value>
ParkeePriority: <value>
ParkeeUniqueid: <value>
ParkerChannel: <value>
ParkerChannelState: <value>
ParkerChannelStateDesc: <value>
ParkerCallerIDNum: <value>
ParkerCallerIDName: <value>
ParkerConnectedLineNum: <value>
ParkerConnectedLineName: <value>
ParkerAccountCode: <value>
ParkerContext: <value>
ParkerExten: <value>
ParkerPriority: <value>
ParkerUniqueid: <value>
ParkerDialString: <value>
Parkinglot: <value>
ParkingSpace: <value>
ParkingTimeout: <value>
ParkingDuration: <value>
RetrieverChannel: <value>
RetrieverChannelState: <value>
RetrieverChannelStateDesc: <value>
RetrieverCallerIDNum: <value>
RetrieverCallerIDName: <value>
RetrieverConnectedLineNum: <value>
RetrieverConnectedLineName: <value>
RetrieverAccountCode: <value>
RetrieverContext: <value>
RetrieverExten: <value>
RetrieverPriority: <value>
RetrieverUniqueid: <value>
```

Arguments

- ParkeeChannel
- ParkeeChannelState - A numeric code for the channel's current state, related to ParkeeChannelStateDesc
- ParkeeChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- ParkeeCallerIDNum
- ParkeeCallerIDName

- ParkeeConnectedLineNum
- ParkeeConnectedLineName
- ParkeeAccountCode
- ParkeeContext
- ParkeeExten
- ParkeePriority
- ParkeeUniqueid
- ParkerChannel
- ParkerChannelState - A numeric code for the channel's current state, related to ParkerChannelStateDesc
- ParkerChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- ParkerCallerIDNum
- ParkerCallerIDName
- ParkerConnectedLineNum
- ParkerConnectedLineName
- ParkerAccountCode
- ParkerContext
- ParkerExten
- ParkerPriority
- ParkerUniqueid
- ParkerDialString - Dial String that can be used to call back the parkee on ParkingTimeout.
- Parkinglot - Name of the parking lot that the parkee is parked in
- ParkingSpace - Parking Space that the parkee is parked in
- ParkingTimeout - Time remaining until the parkee is forcefully removed from parking in seconds
- ParkingDuration - Time the parkee has been in the parking bridge (in seconds)
- RetrieverChannel
- RetrieverChannelState - A numeric code for the channel's current state, related to RetrieverChannelStateDesc
- RetrieverChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- RetrieverCallerIDNum
- RetrieverCallerIDName
- RetrieverConnectedLineNum
- RetrieverConnectedLineName
- RetrieverAccountCode
- RetrieverContext
- RetrieverExten
- RetrieverPriority
- RetrieverUniqueid

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_UserEvent

UserEvent

Synopsis

A user defined event raised from the dialplan.

Description

Syntax

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

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- UserEvent - The event name, as specified in the dialplan.

See Also

- [Asterisk 12 Application_UserEvent](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ManagerEvent_VarSet

VarSet

Synopsis

Raised when a variable local to the gosub stack frame is set due to a subroutine call.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Variable: <value>
Value: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Variable - The LOCAL variable being set.



Note

The variable name will always be enclosed with LOCAL ()

- Value - The new value of the variable.

See Also

- [Asterisk 12 Application_GoSub](#)
- [Asterisk 12 AGICommand_gosub](#)
- [Asterisk 12 Function_LOCAL](#)

- [Asterisk 12 Function_LOCAL_PEEK](#)

Synopsis

Raised when a variable is shared between channels.

Description

Syntax

```
Action:
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Variable: <value>
Value: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Variable - The SHARED variable being set.



Note

The variable name will always be enclosed with SHARED()

- Value - The new value of the variable.

See Also

- [Asterisk 12 Function_SHARED](#)

Synopsis

Raised when a variable is set to a particular value.

Description

Syntax

```
Action:  
Channel: <value>  
ChannelState: <value>  
ChannelStateDesc: <value>  
CallerIDNum: <value>  
CallerIDName: <value>  
ConnectedLineNum: <value>  
ConnectedLineName: <value>  
AccountCode: <value>  
Context: <value>  
Exten: <value>  
Priority: <value>  
Uniqueid: <value>  
Variable: <value>  
Value: <value>
```

Arguments

- Channel
- ChannelState - A numeric code for the channel's current state, related to ChannelStateDesc
- ChannelStateDesc
 - Down
 - Rsrvd
 - OffHook
 - Dialing
 - Ring
 - Ringing
 - Up
 - Busy
 - Dialing Offhook
 - Pre-ring
 - Unknown
- CallerIDNum
- CallerIDName
- ConnectedLineNum
- ConnectedLineName
- AccountCode
- Context
- Exten
- Priority
- Uniqueid
- Variable - The variable being set.
- Value - The new value of the variable.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 ARI

Asterisk 12 introduces the Asterisk REST Interface (ARI), a set of RESTful API's for building Asterisk based applications.

For a brief introduction to ARI, see our [getting started guide](#). See the pages below for a complete reference of the API.

- [Asterisk 12 REST Data Models](#)
- [Asterisk 12 Asterisk REST API](#)
- [Asterisk 12 Bridges REST API](#)
- [Asterisk 12 Channels REST API](#)
- [Asterisk 12 Endpoints REST API](#)
- [Asterisk 12 Events REST API](#)
- [Asterisk 12 Recordings REST API](#)
- [Asterisk 12 Sounds REST API](#)
- [Asterisk 12 Applications REST API](#)
- [Asterisk 12 Playbacks REST API](#)
- [Asterisk 12 Devicestates REST API](#)

Asterisk 12 REST Data Models

- AsteriskInfo
- BuildInfo
- ConfigInfo
- SetId
- StatusInfo
- SystemInfo
- Variable
- Endpoint
- CallerID
- Channel
- Dialed
- DialplanCEP
- Bridge
- LiveRecording
- StoredRecording
- FormatLangPair
- Sound
- Playback
- DeviceState
- ApplicationReplaced
- BridgeCreated
- BridgeDestroyed
- BridgeMerged
- ChannelCallerId
- ChannelCreated
- ChannelDestroyed
- ChannelDialplan
- ChannelDtmfReceived
- ChannelEnteredBridge
- ChannelHangupRequest
- ChannelLeftBridge
- ChannelStateChange
- ChannelUserEvent
- ChannelVarset
- DeviceStateChanged
- Dial
- EndpointStateChange
- Event
- Message
- MissingParams
- PlaybackFinished
- PlaybackStarted
- RecordingFailed
- RecordingFinished
- RecordingStarted
- StasisEnd
- StasisStart
- Application

AsteriskInfo

Asterisk system information

```

{
  "properties": {
    "status": {
      "required": false,
      "type": "StatusInfo",
      "description": "Info about Asterisk status"
    },
    "config": {
      "required": false,
      "type": "ConfigInfo",
      "description": "Info about Asterisk configuration"
    },
    "build": {
      "required": false,
      "type": "BuildInfo",
      "description": "Info about how Asterisk was built"
    },
    "system": {
      "required": false,
      "type": "SystemInfo",
      "description": "Info about the system running Asterisk"
    }
  },
  "id": "AsteriskInfo",
  "description": "Asterisk system information"
}

```

- build: [BuildInfo](#) (*optional*) - Info about how Asterisk was built
- config: [ConfigInfo](#) (*optional*) - Info about Asterisk configuration
- status: [StatusInfo](#) (*optional*) - Info about Asterisk status
- system: [SystemInfo](#) (*optional*) - Info about the system running Asterisk

BuildInfo

Info about how Asterisk was built

```
{
  "properties": {
    "kernel": {
      "required": true,
      "type": "string",
      "description": "Kernel version Asterisk was built on."
    },
    "machine": {
      "required": true,
      "type": "string",
      "description": "Machine architecture (x86_64, i686, ppc, etc.)"
    },
    "user": {
      "required": true,
      "type": "string",
      "description": "Username that build Asterisk"
    },
    "date": {
      "required": true,
      "type": "string",
      "description": "Date and time when Asterisk was built."
    },
    "os": {
      "required": true,
      "type": "string",
      "description": "OS Asterisk was built on."
    },
    "options": {
      "required": true,
      "type": "string",
      "description": "Compile time options, or empty string if default."
    }
  },
  "id": "BuildInfo",
  "description": "Info about how Asterisk was built"
}
```

- date: string - Date and time when Asterisk was built.
- kernel: string - Kernel version Asterisk was built on.
- machine: string - Machine architecture (x86_64, i686, ppc, etc.)
- options: string - Compile time options, or empty string if default.
- os: string - OS Asterisk was built on.
- user: string - Username that build Asterisk

ConfigInfo

Info about Asterisk configuration

```

{
  "properties": {
    "name": {
      "required": true,
      "type": "string",
      "description": "Asterisk system name."
    },
    "default_language": {
      "required": true,
      "type": "string",
      "description": "Default language for media playback."
    },
    "max_load": {
      "required": false,
      "type": "double",
      "description": "Maximum load avg on system."
    },
    "setid": {
      "required": true,
      "type": "SetId",
      "description": "Effective user/group id for running Asterisk."
    },
    "max_open_files": {
      "required": false,
      "type": "int",
      "description": "Maximum number of open file handles (files, sockets).",
    },
    "max_channels": {
      "required": false,
      "type": "int",
      "description": "Maximum number of simultaneous channels."
    }
  },
  "id": "ConfigInfo",
  "description": "Info about Asterisk configuration"
}

```

- default_language: string - Default language for media playback.
- max_channels: int (*optional*) - Maximum number of simultaneous channels.
- max_load: double (*optional*) - Maximum load avg on system.
- max_open_files: int (*optional*) - Maximum number of open file handles (files, sockets).
- name: string - Asterisk system name.
- setid: [SetId](#) - Effective user/group id for running Asterisk.

SetId

Effective user/group id

```
{
  "properties": {
    "group": {
      "required": true,
      "type": "string",
      "description": "Effective group id."
    },
    "user": {
      "required": true,
      "type": "string",
      "description": "Effective user id."
    }
  },
  "id": "SetId",
  "description": "Effective user/group id"
}
```

- group: string - Effective group id.
- user: string - Effective user id.

StatusInfo

Info about Asterisk status

```
{
  "properties": {
    "last_reload_time": {
      "required": true,
      "type": "Date",
      "description": "Time when Asterisk was last reloaded."
    },
    "startup_time": {
      "required": true,
      "type": "Date",
      "description": "Time when Asterisk was started."
    }
  },
  "id": "StatusInfo",
  "description": "Info about Asterisk status"
}
```

- last_reload_time: Date - Time when Asterisk was last reloaded.
- startup_time: Date - Time when Asterisk was started.

SystemInfo

Info about Asterisk

```
{
  "properties": {
    "entity_id": {
      "required": true,
      "type": "string",
      "description": ""
    },
    "version": {
      "required": true,
      "type": "string",
      "description": "Asterisk version."
    }
  },
  "id": "SystemInfo",
  "description": "Info about Asterisk"
}
```

- entity_id: string
- version: string - Asterisk version.

Variable

The value of a channel variable

```
{
  "properties": {
    "value": {
      "required": true,
      "type": "string",
      "description": "The value of the variable requested"
    }
  },
  "id": "Variable",
  "description": "The value of a channel variable"
}
```

- value: string - The value of the variable requested

Endpoint

An external device that may offer/accept calls to/from Asterisk.

Unlike most resources, which have a single unique identifier, an endpoint is uniquely identified by the technology/resource pair.


```

{
  "properties": {
    "resource": {
      "required": true,
      "type": "string",
      "description": "Identifier of the endpoint, specific to the given technology."
    },
    "state": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "unknown",
          "offline",
          "online"
        ]
      },
      "required": false,
      "type": "string",
      "description": "Endpoint's state"
    },
    "technology": {
      "required": true,
      "type": "string",
      "description": "Technology of the endpoint"
    },
    "channel_ids": {
      "required": true,
      "type": "List[string]",
      "description": "Id's of channels associated with this endpoint"
    }
  },
  "id": "Endpoint",
  "description": "An external device that may offer/accept calls to/from Asterisk.\n\nUnlike most resources, which have a single unique identifier, an endpoint is uniquely identified by the technology/resource pair."
}

```

- channel_ids: List[string] - Id's of channels associated with this endpoint
- resource: string - Identifier of the endpoint, specific to the given technology.
- state: string (*optional*) - Endpoint's state
- technology: string - Technology of the endpoint

CallerID

Caller identification

```
{
  "properties": {
    "name": {
      "required": true,
      "type": "string"
    },
    "number": {
      "required": true,
      "type": "string"
    }
  },
  "id": "CallerID",
  "description": "Caller identification"
}
```

- name: string
- number: string

Channel

A specific communication connection between Asterisk and an Endpoint.

```
{
  "properties": {
    "accountcode": {
      "required": true,
      "type": "string"
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Name of the channel (i.e. SIP/foo-0000a7e3)"
    },
    "caller": {
      "required": true,
      "type": "CallerID"
    },
    "creationtime": {
      "required": true,
      "type": "Date",
      "description": "Timestamp when channel was created"
    },
    "state": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "Down",
          "Rsrvd",
          "OffHook",
          "Dialing",
          "Ring",
          "Ringing",
          "Up",
          "Busy",
          "Dialing Offhook",
          "Pre-ring",

```

```
        "Unknown"
      ]
    },
    "required": true,
    "type": "string"
  },
  "connected": {
    "required": true,
    "type": "CallerID"
  },
  "dialplan": {
    "required": true,
    "type": "DialplanCEP",
    "description": "Current location in the dialplan"
  },
  "id": {
    "required": true,
    "type": "string",
    "description": "Unique identifier of the channel.\n\nThis is the same as the Uniqueid field in AMI."
  }
},
"id": "Channel",
```

```
{
  "description": "A specific communication connection between Asterisk and an Endpoint."
}
```

- accountcode: string
- caller: [CallerID](#)
- connected: [CallerID](#)
- creationtime: Date - Timestamp when channel was created
- dialplan: [DialplanCEP](#) - Current location in the dialplan
- id: string - Unique identifier of the channel.

This is the same as the Uniqueid field in AML.

- name: string - Name of the channel (i.e. SIP/foo-0000a7e3)
- state: string

Dialed

Dialed channel information.

```
{
  "properties": {},
  "id": "Dialed",
  "description": "Dialed channel information."
}
```

DialplanCEP

Dialplan location (context/extension/priority)

```
{
  "properties": {
    "priority": {
      "required": true,
      "type": "long",
      "description": "Priority in the dialplan"
    },
    "exten": {
      "required": true,
      "type": "string",
      "description": "Extension in the dialplan"
    },
    "context": {
      "required": true,
      "type": "string",
      "description": "Context in the dialplan"
    }
  },
  "id": "DialplanCEP",
  "description": "Dialplan location (context/extension/priority)"
}
```

- context: string - Context in the dialplan
- exten: string - Extension in the dialplan
- priority: long - Priority in the dialplan

Bridge

The merging of media from one or more channels.

Everyone on the bridge receives the same audio.

```
{
  "properties": {
    "bridge_type": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "mixing",
          "holding"
        ]
      },
      "required": true,
      "type": "string",
      "description": "Type of bridge technology"
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Name the creator gave the bridge"
    },
    "creator": {
      "required": true,
      "type": "string",
      "description": "Entity that created the bridge"
    },
    "channels": {
      "required": true,
      "type": "List[string]",
      "description": "Ids of channels participating in this bridge"
    },
    "bridge_class": {
      "required": true,
      "type": "string",
      "description": "Bridging class"
    },
    "technology": {
      "required": true,
      "type": "string",
      "description": "Name of the current bridging technology"
    },
    "id": {
      "required": true,
      "type": "string",
      "description": "Unique identifier for this bridge"
    }
  },
  "id": "Bridge",
  "description": "The merging of media from one or more channels.\n\nEveryone on the bridge receives the same audio."
}
```

- `bridge_class`: string - Bridging class
- `bridge_type`: string - Type of bridge technology
- `channels`: List[string] - Ids of channels participating in this bridge
- `creator`: string - Entity that created the bridge

- id: string - Unique identifier for this bridge
- name: string - Name the creator gave the bridge
- technology: string - Name of the current bridging technology

LiveRecording

A recording that is in progress

```
{
  "properties": {
    "state": {
      "required": true,
      "type": "string"
    },
    "cause": {
      "required": false,
      "type": "string",
      "description": "Cause for recording failure if failed"
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Base name for the recording"
    },
    "format": {
      "required": true,
      "type": "string"
    }
  },
  "id": "LiveRecording",
  "description": "A recording that is in progress"
}
```

- cause: string (*optional*) - Cause for recording failure if failed
- format: string
- name: string - Base name for the recording
- state: string

StoredRecording

A past recording that may be played back.

```
{
  "properties": {
    "name": {
      "required": true,
      "type": "string"
    },
    "format": {
      "required": true,
      "type": "string"
    }
  },
  "id": "StoredRecording",
  "description": "A past recording that may be played back."
}
```

- format: string
- name: string

FormatLangPair

Identifies the format and language of a sound file

```
{
  "properties": {
    "language": {
      "required": true,
      "type": "string"
    },
    "format": {
      "required": true,
      "type": "string"
    }
  },
  "id": "FormatLangPair",
  "description": "Identifies the format and language of a sound file"
}
```

- format: string
- language: string

Sound

A media file that may be played back.

```
{
  "properties": {
    "text": {
      "required": false,
      "type": "string",
      "description": "Text description of the sound, usually the words spoken."
    },
    "id": {
      "required": true,
      "type": "string",
      "description": "Sound's identifier."
    },
    "formats": {
      "required": true,
      "type": "List[FormatLangPair]",
      "description": "The formats and languages in which this sound is available."
    }
  },
  "id": "Sound",
  "description": "A media file that may be played back."
}
```

- formats: [List\[FormatLangPair\]](#) - The formats and languages in which this sound is available.
- id: string - Sound's identifier.
- text: string (*optional*) - Text description of the sound, usually the words spoken.

Playback

Object representing the playback of media to a channel

```
{
  "properties": {
    "language": {
      "type": "string",
      "description": "For media types that support multiple languages, the language requested for playback."
    },
    "media_uri": {
      "required": true,
      "type": "string",
      "description": "URI for the media to play back."
    },
    "id": {
      "required": true,
      "type": "string",
      "description": "ID for this playback operation"
    },
    "target_uri": {
      "required": true,
      "type": "string",
      "description": "URI for the channel or bridge to play the media on"
    },
    "state": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "queued",
          "playing",
          "complete"
        ]
      },
      "required": true,
      "type": "string",
      "description": "Current state of the playback operation."
    }
  },
  "id": "Playback",
  "description": "Object representing the playback of media to a channel"
}
```

- id: string - ID for this playback operation
- language: string (*optional*) - For media types that support multiple languages, the language requested for playback.
- media_uri: string - URI for the media to play back.
- state: string - Current state of the playback operation.
- target_uri: string - URI for the channel or bridge to play the media on

DeviceState

Represents the state of a device.


```
{
  "properties": {
    "state": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "UNKNOWN",
          "NOT_INUSE",
          "INUSE",
          "BUSY",
          "INVALID",
          "UNAVAILABLE",
          "RINGING",
          "RINGINUSE",
          "ONHOLD"
        ]
      },
      "required": true,
      "type": "string",
      "description": "Device's state"
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Name of the device."
    }
  },
  "id": "DeviceState",
  "description": "Represents the state of a device."
}
```

- name: string - Name of the device.
- state: string - Device's state

ApplicationReplaced

Base type: [Event](#)

Notification that another WebSocket has taken over for an application.

An application may only be subscribed to by a single WebSocket at a time. If multiple WebSockets attempt to subscribe to the same application, the newer WebSocket wins, and the older one receives this event.

```
{
  "properties": {},
  "id": "ApplicationReplaced",
  "description": "Notification that another WebSocket has taken over for an
application.\n\nAn application may only be subscribed to by a single WebSocket at a time.
If multiple WebSockets attempt to subscribe to the same application, the newer WebSocket
wins, and the older one receives this event."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.

BridgeCreated

Base type: [Event](#)

Notification that a bridge has been created.

```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    }
  },
  "id": "BridgeCreated",
  "description": "Notification that a bridge has been created."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- bridge: [Bridge](#)

BridgeDestroyed

Base type: [Event](#)

Notification that a bridge has been destroyed.

```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    }
  },
  "id": "BridgeDestroyed",
  "description": "Notification that a bridge has been destroyed."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- bridge: [Bridge](#)

BridgeMerged

Base type: [Event](#)

Notification that one bridge has merged into another.

```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    },
    "bridge_from": {
      "required": true,
      "type": "Bridge"
    }
  },
  "id": "BridgeMerged",
  "description": "Notification that one bridge has merged into another."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- bridge: [Bridge](#)
- bridge_from: [Bridge](#)

ChannelCallerId

Base type: [Event](#)

Channel changed Caller ID.

```
{
  "properties": {
    "caller_presentation_txt": {
      "required": true,
      "type": "string",
      "description": "The text representation of the Caller Presentation value."
    },
    "caller_presentation": {
      "required": true,
      "type": "int",
      "description": "The integer representation of the Caller Presentation value."
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel that changed Caller ID."
    }
  },
  "id": "ChannelCallerId",
  "description": "Channel changed Caller ID."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- caller_presentation: int - The integer representation of the Caller Presentation value.
- caller_presentation_txt: string - The text representation of the Caller Presentation value.
- channel: [Channel](#) - The channel that changed Caller ID.

ChannelCreated

Base type: [Event](#)

Notification that a channel has been created.

```
{
  "properties": {
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "ChannelCreated",
  "description": "Notification that a channel has been created."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- channel: [Channel](#)

ChannelDestroyed

Base type: [Event](#)

Notification that a channel has been destroyed.

```
{
  "properties": {
    "cause": {
      "required": true,
      "type": "int",
      "description": "Integer representation of the cause of the hangup"
    },
    "cause_txt": {
      "required": true,
      "type": "string",
      "description": "Text representation of the cause of the hangup"
    },
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "ChannelDestroyed",
  "description": "Notification that a channel has been destroyed."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- cause: int - Integer representation of the cause of the hangup
- cause_txt: string - Text representation of the cause of the hangup
- channel: [Channel](#)

ChannelDialplan

Base type: [Event](#)

Channel changed location in the dialplan.

```
{
  "properties": {
    "dialplan_app_data": {
      "required": true,
      "type": "string",
      "description": "The data to be passed to the application."
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel that changed dialplan location."
    },
    "dialplan_app": {
      "required": true,
      "type": "string",
      "description": "The application about to be executed."
    }
  },
  "id": "ChannelDialplan",
  "description": "Channel changed location in the dialplan."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- channel: [Channel](#) - The channel that changed dialplan location.
- dialplan_app: string - The application about to be executed.
- dialplan_app_data: string - The data to be passed to the application.

ChannelDtmfReceived

Base type: [Event](#)

DTMF received on a channel.

This event is sent when the DTMF ends. There is no notification about the start of DTMF

```
{
  "properties": {
    "duration_ms": {
      "required": true,
      "type": "int",
      "description": "Number of milliseconds DTMF was received"
    },
    "digit": {
      "required": true,
      "type": "string",
      "description": "DTMF digit received (0-9, A-E, # or *)"
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel on which DTMF was received"
    }
  },
  "id": "ChannelDtmfReceived",
  "description": "DTMF received on a channel.\n\nThis event is sent when the DTMF ends. There is no notification about the start of DTMF"
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- channel: [Channel](#) - The channel on which DTMF was received
- digit: string - DTMF digit received (0-9, A-E, # or *)
- duration_ms: int - Number of milliseconds DTMF was received

ChannelEnteredBridge

Base type: [Event](#)

Notification that a channel has entered a bridge.

```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    },
    "channel": {
      "type": "Channel"
    }
  },
  "id": "ChannelEnteredBridge",
  "description": "Notification that a channel has entered a bridge."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- bridge: [Bridge](#)
- channel: [Channel](#) (*optional*)

ChannelHangupRequest

Base type: [Event](#)

A hangup was requested on the channel.

```
{
  "properties": {
    "soft": {
      "type": "boolean",
      "description": "Whether the hangup request was a soft hangup request."
    },
    "cause": {
      "type": "int",
      "description": "Integer representation of the cause of the hangup."
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel on which the hangup was requested."
    }
  },
  "id": "ChannelHangupRequest",
  "description": "A hangup was requested on the channel."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- cause: int (*optional*) - Integer representation of the cause of the hangup.
- channel: [Channel](#) - The channel on which the hangup was requested.
- soft: boolean (*optional*) - Whether the hangup request was a soft hangup request.

ChannelLeftBridge

Base type: [Event](#)

Notification that a channel has left a bridge.

```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    },
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "ChannelLeftBridge",
  "description": "Notification that a channel has left a bridge."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- bridge: [Bridge](#)

- channel: [Channel](#)

ChannelStateChange

Base type: [Event](#)

Notification of a channel's state change.

```
{
  "properties": {
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "ChannelStateChange",
  "description": "Notification of a channel's state change."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- channel: [Channel](#)

ChannelUserevent

Base type: [Event](#)

User-generated event with additional user-defined fields in the object.

```
{
  "properties": {
    "eventname": {
      "required": true,
      "type": "string",
      "description": "The name of the user event."
    },
    "userevent": {
      "required": true,
      "type": "object",
      "description": "Custom Userevent data"
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel that signaled the user event."
    }
  },
  "id": "ChannelUserevent",
  "description": "User-generated event with additional user-defined fields in the object."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- channel: [Channel](#) - The channel that signaled the user event.
- eventname: string - The name of the user event.

- userevent: [object](#) - Custom Userevent data

ChannelVarset

Base type: [Event](#)

Channel variable changed.

```
{
  "properties": {
    "variable": {
      "required": true,
      "type": "string",
      "description": "The variable that changed."
    },
    "channel": {
      "required": false,
      "type": "Channel",
      "description": "The channel on which the variable was set.\n\nIf missing, the variable is a global variable."
    },
    "value": {
      "required": true,
      "type": "string",
      "description": "The new value of the variable."
    }
  },
  "id": "ChannelVarset",
  "description": "Channel variable changed."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- channel: [Channel](#) (*optional*) - The channel on which the variable was set.

If missing, the variable is a global variable.

- value: string - The new value of the variable.
- variable: string - The variable that changed.

DeviceStateChanged

Base type: [Event](#)

Notification that a device state has changed.

```
{
  "properties": {
    "device_state": {
      "required": true,
      "type": "DeviceState",
      "description": "Device state object"
    }
  },
  "id": "DeviceStateChanged",
  "description": "Notification that a device state has changed."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- device_state: [DeviceState](#) - Device state object

Dial

Base type: [Event](#)

Dialing state has changed.

```
{
  "properties": {
    "forwarded": {
      "required": false,
      "type": "Channel",
      "description": "Channel that the caller has been forwarded to."
    },
    "caller": {
      "required": false,
      "type": "Channel",
      "description": "The calling channel."
    },
    "dialstatus": {
      "required": true,
      "type": "string",
      "description": "Current status of the dialing attempt to the peer."
    },
    "forward": {
      "required": false,
      "type": "string",
      "description": "Forwarding target requested by the original dialed channel."
    },
    "dialstring": {
      "required": false,
      "type": "string",
      "description": "The dial string for calling the peer channel."
    },
    "peer": {
      "required": true,
      "type": "Channel",
      "description": "The dialed channel."
    }
  },
  "id": "Dial",
  "description": "Dialing state has changed."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- caller: [Channel](#) (*optional*) - The calling channel.
- dialstatus: string - Current status of the dialing attempt to the peer.
- dialstring: string (*optional*) - The dial string for calling the peer channel.
- forward: string (*optional*) - Forwarding target requested by the original dialed channel.
- forwarded: [Channel](#) (*optional*) - Channel that the caller has been forwarded to.
- peer: [Channel](#) - The dialed channel.

EndpointStateChange

Base type: [Event](#)

Endpoint state changed.

```
{
  "properties": {
    "endpoint": {
      "required": true,
      "type": "Endpoint"
    }
  },
  "id": "EndpointStateChange",
  "description": "Endpoint state changed."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- endpoint: [Endpoint](#)

Event

Base type: [Message](#)

Subtypes: [ApplicationReplaced](#) [BridgeCreated](#) [BridgeDestroyed](#) [BridgeMerged](#) [ChannelCallerId](#) [ChannelCreated](#) [ChannelDestroyed](#) [ChannelDialplan](#) [ChannelDtmfReceived](#) [ChannelEnteredBridge](#) [ChannelHangupRequest](#) [ChannelLeftBridge](#) [ChannelStateChange](#) [ChannelUserEvent](#) [ChannelVarset](#) [DeviceStateChanged](#) [Dial](#) [EndpointStateChange](#) [PlaybackFinished](#) [PlaybackStarted](#) [RecordingFailed](#) [RecordingFinished](#) [RecordingStarted](#) [StasisEnd](#) [StasisStart](#)

Base type for asynchronous events from Asterisk.

```

{
  "subTypes": [
    "DeviceStateChanged",
    "PlaybackStarted",
    "PlaybackFinished",
    "RecordingStarted",
    "RecordingFinished",
    "RecordingFailed",
    "ApplicationReplaced",
    "BridgeCreated",
    "BridgeDestroyed",
    "BridgeMerged",
    "ChannelCreated",
    "ChannelDestroyed",
    "ChannelEnteredBridge",
    "ChannelLeftBridge",
    "ChannelStateChange",
    "ChannelDtmfReceived",
    "ChannelDialplan",
    "ChannelCallerId",
    "ChannelUserEvent",
    "ChannelHangupRequest",
    "ChannelVarset",
    "EndpointStateChange",
    "Dial",
    "StasisEnd",
    "StasisStart"
  ],
  "properties": {
    "application": {
      "required": true,
      "type": "string",
      "description": "Name of the application receiving the event."
    },
    "timestamp": {
      "required": false,
      "type": "Date",
      "description": "Time at which this event was created."
    }
  },
  "id": "Event",
  "description": "Base type for asynchronous events from Asterisk."
}

```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.

Message

Subtypes: [ApplicationReplaced](#) [BridgeCreated](#) [BridgeDestroyed](#) [BridgeMerged](#) [ChannelCallerId](#) [ChannelCreated](#) [ChannelDestroyed](#) [ChannelDialplan](#) [ChannelDtmfReceived](#) [ChannelEnteredBridge](#) [ChannelHangupRequest](#) [ChannelLeftBridge](#) [ChannelStateChange](#) [ChannelUserEvent](#) [ChannelVarset](#) [DeviceStateChanged](#) [Dial](#) [EndpointStateChange](#) [Event](#) [MissingParams](#) [PlaybackFinished](#) [PlaybackStarted](#) [RecordingFailed](#) [RecordingFinished](#) [RecordingStarted](#) [StasisEnd](#) [StasisStart](#)

Base type for errors and events

```
{
  "discriminator": "type",
  "properties": {
    "type": {
      "required": true,
      "type": "string",
      "description": "Indicates the type of this message."
    }
  },
  "subTypes": [
    "MissingParams",
    "Event"
  ],
  "id": "Message",
  "description": "Base type for errors and events"
}
```

- type: string - Indicates the type of this message.

MissingParams

Base type: [Message](#)

Error event sent when required params are missing.

```
{
  "properties": {
    "params": {
      "required": true,
      "type": "List[string]",
      "description": "A list of the missing parameters"
    }
  },
  "id": "MissingParams",
  "description": "Error event sent when required params are missing."
}
```

- type: string - Indicates the type of this message.
- params: List[string] - A list of the missing parameters

PlaybackFinished

Base type: [Event](#)

Event showing the completion of a media playback operation.

```
{
  "properties": {
    "playback": {
      "required": true,
      "type": "Playback",
      "description": "Playback control object"
    }
  },
  "id": "PlaybackFinished",
  "description": "Event showing the completion of a media playback operation."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- playback: [Playback](#) - Playback control object

PlaybackStarted

Base type: [Event](#)

Event showing the start of a media playback operation.

```
{
  "properties": {
    "playback": {
      "required": true,
      "type": "Playback",
      "description": "Playback control object"
    }
  },
  "id": "PlaybackStarted",
  "description": "Event showing the start of a media playback operation."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- playback: [Playback](#) - Playback control object

RecordingFailed

Base type: [Event](#)

Event showing failure of a recording operation.

```
{
  "properties": {
    "recording": {
      "required": true,
      "type": "LiveRecording",
      "description": "Recording control object"
    }
  },
  "extends": "Event",
  "id": "RecordingFailed",
  "description": "Event showing failure of a recording operation."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- recording: [LiveRecording](#) - Recording control object

RecordingFinished

Base type: [Event](#)

Event showing the completion of a recording operation.

```
{
  "properties": {
    "recording": {
      "required": true,
      "type": "LiveRecording",
      "description": "Recording control object"
    }
  },
  "extends": "Event",
  "id": "RecordingFinished",
  "description": "Event showing the completion of a recording operation."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- recording: [LiveRecording](#) - Recording control object

RecordingStarted

Base type: [Event](#)

Event showing the start of a recording operation.

```
{
  "properties": {
    "recording": {
      "required": true,
      "type": "LiveRecording",
      "description": "Recording control object"
    }
  },
  "extends": "Event",
  "id": "RecordingStarted",
  "description": "Event showing the start of a recording operation."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- recording: [LiveRecording](#) - Recording control object

StasisEnd

Base type: [Event](#)

Notification that a channel has left a Stasis application.

```
{
  "properties": {
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "StasisEnd",
  "description": "Notification that a channel has left a Stasis application."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- channel: [Channel](#)

StasisStart

Base type: [Event](#)

Notification that a channel has entered a Stasis application.


```
{
  "properties": {
    "args": {
      "required": true,
      "type": "List[string]",
      "description": "Arguments to the application"
    },
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "StasisStart",
  "description": "Notification that a channel has entered a Stasis application."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date (*optional*) - Time at which this event was created.
- args: List[string] - Arguments to the application
- channel: [Channel](#)

Application

Details of a Stasis application

```

{
  "properties": {
    "endpoint_ids": {
      "required": true,
      "type": "List[string]",
      "description": "{tech}/{resource} for endpoints subscribed to."
    },
    "channel_ids": {
      "required": true,
      "type": "List[string]",
      "description": "Id's for channels subscribed to."
    },
    "bridge_ids": {
      "required": true,
      "type": "List[string]",
      "description": "Id's for bridges subscribed to."
    },
    "device_names": {
      "required": true,
      "type": "List[string]",
      "description": "Names of the devices subscribed to."
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Name of this application"
    }
  },
  "id": "Application",
  "description": "Details of a Stasis application"
}

```

- bridge_ids: List[string] - Id's for bridges subscribed to.
- channel_ids: List[string] - Id's for channels subscribed to.
- device_names: List[string] - Names of the devices subscribed to.
- endpoint_ids: List[string] - {tech}/{resource} for endpoints subscribed to.
- name: string - Name of this application

Asterisk 12 Asterisk REST API

Asterisk

Method	Path	Return Model	Summary
GET	/asterisk/info	AsteriskInfo	Gets Asterisk system information.
GET	/asterisk/variable	Variable	Get the value of a global variable.
POST	/asterisk/variable	void	Set the value of a global variable.

GET [/asterisk/info](#)

Gets Asterisk system information.

Query parameters

- only: string - Filter information returned
 - Allows comma separated values.

GET [/asterisk/variable](#)

Get the value of a global variable.

Query parameters

- variable: string - **(required)** The variable to get

Error Responses

- 400 - Missing variable parameter.

POST [/asterisk/variable](#)

Set the value of a global variable.

Query parameters

- variable: string - **(required)** The variable to set
- value: string - The value to set the variable to

Error Responses

- 400 - Missing variable parameter.

Asterisk 12 Bridges REST API

Bridges

Method	Path	Return Model	Summary
GET	/bridges	List[Bridge]	List all active bridges in Asterisk.
POST	/bridges	Bridge	Create a new bridge.
GET	/bridges/{bridgeld}	Bridge	Get bridge details.
DELETE	/bridges/{bridgeld}	void	Shut down a bridge.
POST	/bridges/{bridgeld}/addChannel	void	Add a channel to a bridge.
POST	/bridges/{bridgeld}/removeChannel	void	Remove a channel from a bridge.
POST	/bridges/{bridgeld}/moh	void	Play music on hold to a bridge or change the MOH class that is playing.
DELETE	/bridges/{bridgeld}/moh	void	Stop playing music on hold to a bridge.
POST	/bridges/{bridgeld}/play	Playback	Start playback of media on a bridge.
POST	/bridges/{bridgeld}/record	LiveRecording	Start a recording.

GET /bridges

List all active bridges in Asterisk.

POST /bridges

Create a new bridge. This bridge persists until it has been shut down, or Asterisk has been shut down.

Query parameters

- type: string - Type of bridge to create.
- name: string - Name to give to the bridge being created.

GET /bridges/{bridgeld}

Get bridge details.

Path parameters

- bridgeld: string - Bridge's id

Error Responses

- 404 - Bridge not found

DELETE /bridges/{bridgeld}

Shut down a bridge. If any channels are in this bridge, they will be removed and resume whatever they were doing beforehand.

Path parameters

- bridgeld: string - Bridge's id

Error Responses

- 404 - Bridge not found

POST /bridges/{bridgeld}/addChannel

Add a channel to a bridge.

Path parameters

- bridgeld: string - Bridge's id

Query parameters

- channel: string - **(required)** Ids of channels to add to bridge
 - Allows comma separated values.
- role: string - Channel's role in the bridge

Error Responses

- 400 - Channel not found
- 404 - Bridge not found
- 409 - Bridge not in Stasis application; Channel currently recording
- 422 - Channel not in Stasis application

POST /bridges/{bridgeld}/removeChannel

Remove a channel from a bridge.

Path parameters

- bridgeld: string - Bridge's id

Query parameters

- channel: string - **(required)** Ids of channels to remove from bridge
 - Allows comma separated values.

Error Responses

- 400 - Channel not found
- 404 - Bridge not found
- 409 - Bridge not in Stasis application
- 422 - Channel not in this bridge

POST /bridges/{bridgeld}/moh

Play music on hold to a bridge or change the MOH class that is playing.

Path parameters

- bridgeld: string - Bridge's id

Query parameters

- mohClass: string - Channel's id

Error Responses

- 404 - Bridge not found
- 409 - Bridge not in Stasis application

DELETE /bridges/{bridgeld}/moh

Stop playing music on hold to a bridge. This will only stop music on hold being played via POST bridges/{bridgeld}/moh.

Path parameters

- bridgeld: string - Bridge's id

Error Responses

- 404 - Bridge not found
- 409 - Bridge not in Stasis application

POST /bridges/{bridgeld}/play

Start playback of media on a bridge. The media URI may be any of a number of URI's. Currently sound: and recording: URI's are supported. This operation creates a playback resource that can be used to control the playback of media (pause, rewind, fast forward, etc.)

Path parameters

- bridgeld: string - Bridge's id

Query parameters

- media: string - **(required)** Media's URI to play.
- lang: string - For sounds, selects language for sound.
- offsetms: int - Number of media to skip before playing.
- skipms: int = 3000 - Number of milliseconds to skip for forward/reverse operations.

Error Responses

- 404 - Bridge not found
- 409 - Bridge not in a Stasis application

POST /bridges/{bridgeld}/record

Start a recording. This records the mixed audio from all channels participating in this bridge.

Path parameters

- bridgeld: string - Bridge's id

Query parameters

- name: string - **(required)** Recording's filename
- format: string - **(required)** Format to encode audio in
- maxDurationSeconds: int - Maximum duration of the recording, in seconds. 0 for no limit.
- maxSilenceSeconds: int - Maximum duration of silence, in seconds. 0 for no limit.
- ifExists: string = fail - Action to take if a recording with the same name already exists.
- beep: boolean - Play beep when recording begins
- terminateOn: string = none - DTMF input to terminate recording.

Error Responses

- 400 - Invalid parameters
- 404 - Bridge not found
- 409 - Bridge is not in a Stasis application; A recording with the same name already exists on the system and can not be overwritten because it is in progress or ifExists=fail
- 422 - The format specified is unknown on this system

Asterisk 12 Channels REST API

Channels

Method	Path	Return Model	Summary
GET	/channels	List[Channel]	List all active channels in Asterisk.
POST	/channels	Channel	Create a new channel (originate).
GET	/channels/{channelId}	Channel	Channel details.
DELETE	/channels/{channelId}	void	Delete (i.e. hangup) a channel.
POST	/channels/{channelId}/continue	void	Exit application; continue execution in the dialplan.
POST	/channels/{channelId}/answer	void	Answer a channel.
POST	/channels/{channelId}/ring	void	Indicate ringing to a channel.
DELETE	/channels/{channelId}/ring	void	Stop ringing indication on a channel if locally generated.
POST	/channels/{channelId}/dtmf	void	Send provided DTMF to a given channel.
POST	/channels/{channelId}/mute	void	Mute a channel.
DELETE	/channels/{channelId}/mute	void	Unmute a channel.
POST	/channels/{channelId}/hold	void	Hold a channel.
DELETE	/channels/{channelId}/hold	void	Remove a channel from hold.
POST	/channels/{channelId}/moh	void	Play music on hold to a channel.
DELETE	/channels/{channelId}/moh	void	Stop playing music on hold to a channel.
POST	/channels/{channelId}/silence	void	Play silence to a channel.
DELETE	/channels/{channelId}/silence	void	Stop playing silence to a channel.
POST	/channels/{channelId}/play	Playback	Start playback of media.
POST	/channels/{channelId}/record	LiveRecording	Start a recording.
GET	/channels/{channelId}/variable	Variable	Get the value of a channel variable or function.
POST	/channels/{channelId}/variable	void	Set the value of a channel variable or function.
POST	/channels/{channelId}/snoop	Channel	Start snooping.

GET /channels

List all active channels in Asterisk.

POST /channels

Create a new channel (originate). The new channel is created immediately and a snapshot of it returned. If a Stasis application is provided it will be automatically subscribed to the originated channel for further events and updates.

Query parameters

- endpoint: string - **(required)** Endpoint to call.
- extension: string - The extension to dial after the endpoint answers
- context: string - The context to dial after the endpoint answers. If omitted, uses 'default'
- priority: long - The priority to dial after the endpoint answers. If omitted, uses 1
- app: string - The application that is subscribed to the originated channel, and passed to the Stasis application.
- appArgs: string - The application arguments to pass to the Stasis application.
- callerId: string - CallerID to use when dialing the endpoint or extension.
- timeout: int = 30 - Timeout (in seconds) before giving up dialing, or -1 for no timeout.

Error Responses

- 400 - Invalid parameters for originating a channel.

GET /channels/{channelId}

Channel details.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found

DELETE /channels/{channelId}

Delete (i.e. hangup) a channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- reason: string - Reason for hanging up the channel

Error Responses

- 400 - Invalid reason for hangup provided
- 404 - Channel not found

POST /channels/{channelId}/continue

Exit application; continue execution in the dialplan.

Path parameters

- channelId: string - Channel's id

Query parameters

- context: string - The context to continue to.
- extension: string - The extension to continue to.
- priority: int - The priority to continue to.

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/answer

Answer a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/ring

Indicate ringing to a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/ring

Stop ringing indication on a channel if locally generated.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/dtmf

Send provided DTMF to a given channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- dtmf: string - DTMF To send.
- before: int - Amount of time to wait before DTMF digits (specified in milliseconds) start.
- between: int = 100 - Amount of time in between DTMF digits (specified in milliseconds).
- duration: int = 100 - Length of each DTMF digit (specified in milliseconds).
- after: int - Amount of time to wait after DTMF digits (specified in milliseconds) end.

Error Responses

- 400 - DTMF is required
- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/mute

Mute a channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- direction: string = both - Direction in which to mute audio

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/mute

Unmute a channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- direction: string = both - Direction in which to unmute audio

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/hold

Hold a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/hold

Remove a channel from hold.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/moh

Play music on hold to a channel. Using media operations such as /play on a channel playing MOH in this manner will suspend MOH without resuming automatically. If continuing music on hold is desired, the stasis application must reinitiate music on hold.

Path parameters

- channelId: string - Channel's id

Query parameters

- mohClass: string - Music on hold class to use

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/moh

Stop playing music on hold to a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/silence

Play silence to a channel. Using media operations such as /play on a channel playing silence in this manner will suspend silence without resuming automatically.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/silence

Stop playing silence to a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/play

Start playback of media. The media URI may be any of a number of URI's. Currently sound: and recording: URI's are supported. This operation creates a

playback resource that can be used to control the playback of media (pause, rewind, fast forward, etc.)

Path parameters

- channelId: string - Channel's id

Query parameters

- media: string - **(required)** Media's URI to play.
- lang: string - For sounds, selects language for sound.
- offsetms: int - Number of media to skip before playing.
- skipms: int = 3000 - Number of milliseconds to skip for forward/reverse operations.

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/record

Start a recording. Record audio from a channel. Note that this will not capture audio sent to the channel. The bridge itself has a record feature if that's what you want.

Path parameters

- channelId: string - Channel's id

Query parameters

- name: string - **(required)** Recording's filename
- format: string - **(required)** Format to encode audio in
- maxDurationSeconds: int - Maximum duration of the recording, in seconds. 0 for no limit
- maxSilenceSeconds: int - Maximum duration of silence, in seconds. 0 for no limit
- ifExists: string = fail - Action to take if a recording with the same name already exists.
- beep: boolean - Play beep when recording begins
- terminateOn: string = none - DTMF input to terminate recording

Error Responses

- 400 - Invalid parameters
- 404 - Channel not found
- 409 - Channel is not in a Stasis application; the channel is currently bridged with other hchannels; A recording with the same name already exists on the system and can not be overwritten because it is in progress or ifExists=fail
- 422 - The format specified is unknown on this system

GET /channels/{channelId}/variable

Get the value of a channel variable or function.

Path parameters

- channelId: string - Channel's id

Query parameters

- variable: string - **(required)** The channel variable or function to get

Error Responses

- 400 - Missing variable parameter.
- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/variable

Set the value of a channel variable or function.

Path parameters

- channelId: string - Channel's id

Query parameters

- variable: string - **(required)** The channel variable or function to set
- value: string - The value to set the variable to

Error Responses

- 400 - Missing variable parameter.
- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/snoop

Start snooping. Snoop (spy/whisper) on a specific channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- spy: string = none - Direction of audio to spy on
- whisper: string = none - Direction of audio to whisper into
- app: string - **(required)** Application the snooping channel is placed into
- appArgs: string - The application arguments to pass to the Stasis application

Error Responses

- 400 - Invalid parameters
- 404 - Channel not found

Asterisk 12 Endpoints REST API

Endpoints

Method	Path	Return Model	Summary
GET	/endpoints	List[Endpoint]	List all endpoints.
GET	/endpoints/{tech}	List[Endpoint]	List available endpoints for a given endpoint technology.
GET	/endpoints/{tech}/{resource}	Endpoint	Details for an endpoint.

GET /endpoints

List all endpoints.

GET /endpoints/{tech}

List available endpoints for a given endpoint technology.

Path parameters

- tech: string - Technology of the endpoints (sip,iax2,...)

Error Responses

- 404 - Endpoints not found

GET /endpoints/{tech}/{resource}

Details for an endpoint.

Path parameters

- tech: string - Technology of the endpoint
- resource: string - ID of the endpoint

Error Responses

- 404 - Endpoints not found

Asterisk 12 Events REST API

Events

Method	Path	Return Model	Summary
GET	/events	Message	WebSocket connection for events.

GET [/events](#)

WebSocket connection for events.

Query parameters

- app: string - **(required)** Applications to subscribe to.
 - Allows comma separated values.

Asterisk 12 Recordings REST API

Recordings

Method	Path	Return Model	Summary
GET	/recordings/stored	List[StoredRecording]	List recordings that are complete.
GET	/recordings/stored/{recordingName}	StoredRecording	Get a stored recording's details.
DELETE	/recordings/stored/{recordingName}	void	Delete a stored recording.
GET	/recordings/live/{recordingName}	LiveRecording	List live recordings.
DELETE	/recordings/live/{recordingName}	void	Stop a live recording and discard it.
POST	/recordings/live/{recordingName}/stop	void	Stop a live recording and store it.
POST	/recordings/live/{recordingName}/pause	void	Pause a live recording.
DELETE	/recordings/live/{recordingName}/pause	void	Unpause a live recording.
POST	/recordings/live/{recordingName}/mute	void	Mute a live recording.
DELETE	/recordings/live/{recordingName}/mute	void	Unmute a live recording.

GET /recordings/stored

List recordings that are complete.

GET /recordings/stored/{recordingName}

Get a stored recording's details.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

DELETE /recordings/stored/{recordingName}

Delete a stored recording.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

GET /recordings/live/{recordingName}

List live recordings.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

DELETE /recordings/live/{recordingName}

Stop a live recording and discard it.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

POST /recordings/live/{recordingName}/stop

Stop a live recording and store it.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

POST /recordings/live/{recordingName}/pause

Pause a live recording. Pausing a recording suspends silence detection, which will be restarted when the recording is unpaused. Paused time is not included in the accounting for maxDurationSeconds.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found
- 409 - Recording not in session

DELETE /recordings/live/{recordingName}/pause

Unpause a live recording.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found
- 409 - Recording not in session

POST /recordings/live/{recordingName}/mute

Mute a live recording. Muting a recording suspends silence detection, which will be restarted when the recording is unmuted.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found
- 409 - Recording not in session

DELETE /recordings/live/{recordingName}/mute

Unmute a live recording.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found
- 409 - Recording not in session

Asterisk 12 Sounds REST API

Sounds

Method	Path	Return Model	Summary
GET	/sounds	List[Sound]	List all sounds.
GET	/sounds/{soundId}	Sound	Get a sound's details.

GET /sounds

List all sounds.

Query parameters

- lang: string - Lookup sound for a specific language.
- format: string - Lookup sound in a specific format.

GET /sounds/{soundId}

Get a sound's details.

Path parameters

- soundId: string - Sound's id

Asterisk 12 Applications REST API

Applications

Method	Path	Return Model	Summary
GET	/applications	List[Application]	List all applications.
GET	/applications/{applicationName}	Application	Get details of an application.
POST	/applications/{applicationName}/subscription	Application	Subscribe an application to a event source.
DELETE	/applications/{applicationName}/subscription	Application	Unsubscribe an application from an event source.

GET /applications

List all applications.

GET /applications/{applicationName}

Get details of an application.

Path parameters

- applicationName: string - Application's name

Error Responses

- 404 - Application does not exist.

POST /applications/{applicationName}/subscription

Subscribe an application to a event source. Returns the state of the application after the subscriptions have changed

Path parameters

- applicationName: string - Application's name

Query parameters

- eventSource: string - **(required)** URI for event source (channel:{channelId}, bridge:{bridgeId}, endpoint:{tech}/{resource}, deviceState:{deviceName}
 - Allows comma separated values.

Error Responses

- 400 - Missing parameter.
- 404 - Application does not exist.
- 422 - Event source does not exist.

DELETE /applications/{applicationName}/subscription

Unsubscribe an application from an event source. Returns the state of the application after the subscriptions have changed

Path parameters

- applicationName: string - Application's name

Query parameters

- eventSource: string - **(required)** URI for event source (channel:{channelId}, bridge:{bridgeId}, endpoint:{tech}/{resource}, device_state:{deviceName})
 - Allows comma separated values.

Error Responses

- 400 - Missing parameter; event source scheme not recognized.
- 404 - Application does not exist.
- 409 - Application not subscribed to event source.
- 422 - Event source does not exist.

Asterisk 12 Playbacks REST API

Playbacks

Method	Path	Return Model	Summary
GET	/playbacks/{playbackId}	Playback	Get a playback's details.
DELETE	/playbacks/{playbackId}	void	Stop a playback.
POST	/playbacks/{playbackId}/control	void	Control a playback.

GET /playbacks/{playbackId}

Get a playback's details.

Path parameters

- playbackId: string - Playback's id

Error Responses

- 404 - The playback cannot be found

DELETE /playbacks/{playbackId}

Stop a playback.

Path parameters

- playbackId: string - Playback's id

Error Responses

- 404 - The playback cannot be found

POST /playbacks/{playbackId}/control

Control a playback.

Path parameters

- playbackId: string - Playback's id

Query parameters

- operation: string - **(required)** Operation to perform on the playback.

Error Responses

- 400 - The provided operation parameter was invalid
- 404 - The playback cannot be found
- 409 - The operation cannot be performed in the playback's current state

Asterisk 12 Devicestates REST API

Devicestates

Method	Path	Return Model	Summary
GET	/deviceStates	List[DeviceState]	List all ARI controlled device states.
GET	/deviceStates/{deviceName}	DeviceState	Retrieve the current state of a device.
PUT	/deviceStates/{deviceName}	void	Change the state of a device controlled by ARI. (Note - implicitly creates the device state).
DELETE	/deviceStates/{deviceName}	void	Destroy a device-state controlled by ARI.

GET /deviceStates

List all ARI controlled device states.

GET /deviceStates/{deviceName}

Retrieve the current state of a device.

Path parameters

- deviceName: string - Name of the device

PUT /deviceStates/{deviceName}

Change the state of a device controlled by ARI. (Note - implicitly creates the device state).

Path parameters

- deviceName: string - Name of the device

Query parameters

- deviceState: string - **(required)** Device state value

Error Responses

- 404 - Device name is missing
- 409 - Uncontrolled device specified

DELETE /deviceStates/{deviceName}

Destroy a device-state controlled by ARI.

Path parameters

- deviceName: string - Name of the device

Error Responses

- 404 - Device name is missing
- 409 - Uncontrolled device specified

Asterisk 12 Dialplan Applications

Asterisk 12 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 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_GetCPEID](#)
- `adsi.conf`

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_AgentLogin

AgentLogin()

Synopsis

Login an agent.

Description

Login an agent to the system. Any agent authentication is assumed to already be done by dialplan. While logged in, the agent can receive calls and will hear the sound file specified by the config option `custom_beep` when a new call comes in for the agent. Login failures will continue in the dialplan with `AGENT_T_STATUS` set.

Before logging in, you can setup on the real agent channel the `CHANNEL(dtmf-features)` an agent will have when talking to a caller and you can setup on the channel running this application the `CONNECTEDLINE()` information the agent will see while waiting for a caller.

AGENT_STATUS enumeration values:

- `INVALID` - The specified agent is invalid.
- `ALREADY_LOGGED_IN` - The agent is already logged in.



Note

The Agents:*AgentId* device state is available to monitor the status of the agent.

Syntax

```
AgentLogin(AgentId,options)
```

Arguments

- `AgentId`
- `options`
 - `s` - silent login - do not announce the login ok segment after agent logged on.

See Also

- [Asterisk 12 Application_Authenticate](#)
- [Asterisk 12 Application_Queue](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_AGENT](#)
- [Asterisk 12 Function_CHANNEL\(dtmf-features\)](#)
- [Asterisk 12 Function_CONNECTEDLINE\(\)](#)
- `agents.conf`
- `queues.conf`

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_AgentRequest

AgentRequest()

Synopsis

Request an agent to connect with the channel.

Description

Request an agent to connect with the channel. Failure to find and alert an agent will continue in the dialplan with `AGENT_STATUS` set.

`AGENT_STATUS` enumeration values:

- `INVALID` - The specified agent is invalid.
- `NOT_LOGGED_IN` - The agent is not available.
- `BUSY` - The agent is on another call.
- `ERROR` - Alerting the agent failed.

Syntax

```
AgentRequest (AgentId)
```

Arguments

- `AgentId`

See Also

- [Asterisk 12 Application_AgentLogin](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_AGI

AGI()

Synopsis

Executes an AGI compliant application.

Description

Executes an Asterisk Gateway Interface compliant program on a channel. AGI allows Asterisk to launch external programs written in any language to control a telephony channel, play audio, read DTMF digits, etc. by communicating with the AGI protocol on **stdin** and **stdout**. As of 1.6.0, this channel will not stop dialplan execution on hangup inside of this application. Dialplan execution will continue normally, even upon hangup until the AGI application signals a desire to stop (either by exiting or, in the case of a net script, by closing the connection). A locally executed AGI script will receive SIGHUP on hangup from the channel except when using DeadAGI. A fast AGI server will correspondingly receive a HANGUP inline with the command dialog. Both of these signals may be disabled by setting the `AGISIGHUP` channel variable to `no` before executing the AGI application. Alternatively, if you would like the AGI application to exit immediately after a channel hangup is detected, set the `AGIEXITONHANGUP` variable to `yes`.

Use the CLI command `agi show commands` to list available agi commands.

This application sets the following channel variable upon completion:

- `AGISTATUS` - The status of the attempt to the run the AGI script text string, one of:
 - `SUCCESS`
 - `FAILURE`
 - `NOTFOUND`
 - `HANGUP`

Syntax

```
AGI(commandarg1arg2[...])
```

Arguments

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

See Also

- [Asterisk 12 Application_EAGI](#)
- [Asterisk 12 Application_DeadAGI](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_AlarmReceiver

AlarmReceiver()

Synopsis

Provide support for receiving alarm reports from a burglar or fire alarm panel.

Description

This application should be called whenever there is an alarm panel calling in to dump its events. The application will handshake with the alarm panel, and receive events, validate them, handshake them, and store them until the panel hangs up. Once the panel hangs up, the application will run the system command specified by the eventcmd setting in `alarmreceiver.conf` and pipe the events to the standard input of the application. The configuration file also contains settings for DTMF timing, and for the loudness of the acknowledgement tones.



Note

Few Ademco DTMF signalling formats are detected automatically: Contact ID, Express 4+1, Express 4+2, High Speed and Super Fast.

The application is affected by the following variables:

- `ALARMRECEIVER_CALL_LIMIT` - Maximum call time, in milliseconds.
If set, this variable causes application to exit after the specified time.
- `ALARMRECEIVER_RETRIES_LIMIT` - Maximum number of retries per call.
If set, this variable causes application to exit after the specified number of messages.

Syntax

```
AlarmReceiver()
```

Arguments

See Also

- `alarmreceiver.conf`

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_AMD

AMD()

Synopsis

Attempt to detect answering machines.

Description

This application attempts to detect answering machines at the beginning of outbound calls. Simply call this application after the call has been answered (outbound only, of course).

When loaded, AMD reads amd.conf and uses the parameters specified as default values. Those default values get overwritten when the calling AMD with parameters.

This application sets the following channel variables:

- **AMDSTATUS** - This is the status of the answering machine detection
 - MACHINE
 - HUMAN
 - NOTSURE
 - HANGUP
- **AMDCAUSE** - Indicates the cause that led to the conclusion
 - TOOLONG - Total Time.
 - INITIALSILENCE - Silence Duration - Initial Silence.
 - HUMAN - Silence Duration - afterGreetingSilence.
 - LONGGREETING - Voice Duration - Greeting.
 - MAXWORDLENGTH - Word Count - maximum number of words.

Syntax

```
AMD([initialSilence[,greeting[,afterGreetingSilence[,totalAnalysis  
Time[,miniumWordLength[,betweenWordSilence[,maximumNumberOfWords[,silenceThreshold[,maximumWordLength]]]]]]]])
```

Arguments

- **initialSilence** - Is maximum initial silence duration before greeting.
If this is exceeded set as MACHINE
- **greeting** - is the maximum length of a greeting.
If this is exceeded set as MACHINE
- **afterGreetingSilence** - Is the silence after detecting a greeting.
If this is exceeded set as HUMAN
- **totalAnalysis Time** - Is the maximum time allowed for the algorithm
to decide HUMAN or MACHINE
- **miniumWordLength** - Is the minimum duration of Voice considered to be a word
- **betweenWordSilence** - Is the minimum duration of silence after a word to consider the audio that follows to be a new word
- **maximumNumberOfWords** - Is the maximum number of words in a greeting
If this is exceeded set as MACHINE
- **silenceThreshold** - How long do we consider silence
- **maximumWordLength** - Is the maximum duration of a word to accept.
If exceeded set as MACHINE

See Also

- [Asterisk 12 Application_WaitForSilence](#)
- [Asterisk 12 Application_WaitForNoise](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Answer

Answer()

Synopsis

Answer a channel if ringing.

Description

If the call has not been answered, this application will answer it. Otherwise, it has no effect on the call.

Syntax

```
Answer(delay)
```

Arguments

- `delay` - Asterisk will wait this number of milliseconds before returning to the dialplan after answering the call.

See Also

- [Asterisk 12 Application_Hangup](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404320

Asterisk 12 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 12 Application_VMAuthenticate](#)
- [Asterisk 12 Application_DISA](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_BackGround

BackGround()

Synopsis

Play an audio file while waiting for digits of an extension to go to.

Description

This application will play the given list of files (**do not put extension**) while waiting for an extension to be dialed by the calling channel. To continue waiting for digits after this application has finished playing files, the `WaitExten` application should be used.

If one of the requested sound files does not exist, call processing will be terminated.

This application sets the following channel variable upon completion:

- `BACKGROUNDSTATUS` - The status of the background attempt as a text string.
 - `SUCCESS`
 - `FAILED`

Syntax

```
BackGround(filename1&filename2[&...],options,langoverride,context)
```

Arguments

- `filenames`
 - `filename1`
 - `filename2`
- `options`
 - `s` - Causes the playback of the message to be skipped if the channel is not in the `up` state (i.e. it hasn't been answered yet). If this happens, the application will return immediately.
 - `n` - Don't answer the channel before playing the files.
 - `m` - Only break if a digit hit matches a one digit extension in the destination context.
- `langoverride` - Explicitly specifies which language to attempt to use for the requested sound files.
- `context` - This is the dialplan context that this application will use when exiting to a dialed extension.

See Also

- [Asterisk 12 Application_ControlPlayback](#)
- [Asterisk 12 Application_WaitExten](#)
- [Asterisk 12 Application_BackgroundDetect](#)
- [Asterisk 12 Function_TIMEOUT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_BackgroundDetect

BackgroundDetect()

Synopsis

Background a file with talk detect.

Description

Plays back *filename*, waiting for interruption from a given digit (the digit must start the beginning of a valid extension, or it will be ignored). During the playback of the file, audio is monitored in the receive direction, and if a period of non-silence which is greater than *min* ms yet less than *max* ms is followed by silence for at least *sil* ms, which occurs during the first *analysistime* ms, then the audio playback is aborted and processing jumps to the *talk* extension, if available.

Syntax

```
BackgroundDetect(filename,sil,min,max,analysistime)
```

Arguments

- *filename*
- *sil* - If not specified, defaults to 1000.
- *min* - If not specified, defaults to 100.
- *max* - If not specified, defaults to infinity.
- *analysistime* - If not specified, defaults to infinity.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Bridge

Bridge()

Synopsis

Bridge two channels.

Description

Allows the ability to bridge two channels via the dialplan.


This application sets the following channel variable upon completion:

- **BRIDGERESULT** - The result of the bridge attempt as a text string.
 - **SUCCESS**
 - **FAILURE**
 - **LOOP**
 - **NONEXISTENT**
 - **INCOMPATIBLE**

Syntax

```
Bridge(channel,options)
```

Arguments

- **channel** - The current channel is bridged to the specified *channel*.
- **options**
 - **p** - Play a courtesy tone to *channel*.
 - **F** - When the bridger hangs up, transfer the **bridged** party to the specified destination and **start** execution at that location.
 - **context**
 - **exten**
 - **priority**
 - **F** - When the bridger hangs up, transfer the **bridged** party to the next priority of the current extension and **start** execution at that location.
 - **h** - Allow the called party to hang up by sending the *DTMF digit.
 - **H** - Allow the calling party to hang up by pressing the *DTMF digit.
 - **k** - Allow the called party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.conf`.
 - **K** - Allow the calling party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.conf`.
 - **L(xyz)** - Limit the call to x ms. Play a warning when y ms are left. Repeat the warning every z ms. The following special variables can be used with this option:
 - **LIMIT_PLAYAUDIO_CALLER** - Play sounds to the caller. yes|no (default yes)
 - **LIMIT_PLAYAUDIO_CALLEE** - Play sounds to the callee. yes|no
 - **LIMIT_TIMEOUT_FILE** - File to play when time is up.
 - **LIMIT_CONNECT_FILE** - File to play when call begins.
 - **LIMIT_WARNING_FILE** - File to play as warning if y is defined. The default is to say the time remaining.
 - **S**  - Hang up the call after x seconds **after** the called party has answered the call.
 - **t** - Allow the called party to transfer the calling party by sending the DTMF sequence defined in `features.conf`.
 - **T** - Allow the calling party to transfer the called party by sending the DTMF sequence defined in `features.conf`.
 - **w** - Allow the called party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in `features.conf`.
 - **w** - Allow the calling party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in `features.conf`.
 - **x** - Cause the called party to be hung up after the bridge, instead of being restarted in the dialplan.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_BridgeWait

BridgeWait()

Synopsis

Put a call into the holding bridge.

Description

This application places the incoming channel into a holding bridge. The channel will then wait in the holding bridge until some event occurs which removes it from the holding bridge.



Note

This application will answer calls which haven't already been answered.

Syntax

```
BridgeWait(name[,role,options])
```

Arguments

- **name** - Name of the holding bridge to join. This is a handle for `BridgeWait` only and does not affect the actual bridges that are created. If not provided, the reserved name `default` will be used.
- **role** - Defines the channel's purpose for entering the holding bridge. Values are case sensitive.
 - **participant** - The channel will enter the holding bridge to be placed on hold until it is removed from the bridge for some reason. (default)
 - **announcer** - The channel will enter the holding bridge to make announcements to channels that are currently in the holding bridge. While an announcer is present, holding for the participants will be suspended.
- **options**
 - **m** - The specified MOH class will be used/suggested for music on hold operations. This option will only be useful for entertainment modes that use it (m and h).
 - **class**
 - **e** - Which entertainment mechanism should be used while on hold in the holding bridge. Only the first letter is read.
 - **m** - Play music on hold (default)
 - **r** - Ring without pause
 - **s** - Generate silent audio
 - **h** - Put the channel on hold
 - **n** - No entertainment
 - **s** - Automatically exit the bridge and return to the PBX after **duration** seconds.
 - **duration**

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Busy

Busy()

Synopsis

Indicate the Busy condition.

Description

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

Syntax

```
Busy(timeout)
```

Arguments

- `timeout` - If specified, the calling channel will be hung up after the specified number of seconds. Otherwise, this application will wait until the calling channel hangs up.

See Also

- [Asterisk 12 Application_Congestion](#)
- [Asterisk 12 Application_Progress](#)
- [Asterisk 12 Application_Playtones](#)
- [Asterisk 12 Application_Hangup](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_CallCompletionCancel

CallCompletionCancel()

Synopsis

Cancel call completion service

Description

Cancel a Call Completion Request.

This application sets the following channel variables:

- CC_CANCEL_RESULT - This is the returned status of the cancel.
 - SUCCESS
 - FAIL
- CC_CANCEL_REASON - This is the reason the cancel failed.
 - NO_CORE_INSTANCE
 - NOT_GENERIC
 - UNSPECIFIED

Syntax

```
CallCompletionCancel()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_CELGenUserEvent

CELGenUserEvent()

Synopsis

Generates a CEL User Defined Event.

Description

A CEL event will be immediately generated by this channel, with the supplied name for a type.

Syntax

```
CELGenUserEvent(event-name[extra])
```

Arguments

- event-name
 - event-name
 - extra - Extra text to be included with the event.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_ChansAvail

ChansAvail()

Synopsis

Check channel availability

Description

This application will check to see if any of the specified channels are available.

This application sets the following channel variables:

- `AVAILCHAN` - The name of the available channel, if one exists
- `AVAILORIGCHAN` - The canonical channel name that was used to create the channel
- `AVAILSTATUS` - The device state for the device
- `AVAILCAUSECODE` - The cause code returned when requesting the channel

Syntax

```
ChanIsAvail(Technology2/Resource2[&...][,options])
```

Arguments

- `Technology/Resource` - ** `Technology2/Resource2` - Optional extra devices to check
If you need more then one enter them as `Technology2/Resource2&Technology3/Resource3&.....`
Specification of the device(s) to check. These must be in the format of `Technology/Resource`, where *Technology* represents a particular channel driver, and *Resource* represents a resource available to that particular channel driver.
- `options`
 - `a` - Check for all available channels, not only the first one
 - `s` - Consider the channel unavailable if the channel is in use at all
 - `t` - Simply checks if specified channels exist in the channel list

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ChannelRedirect

ChannelRedirect()

Synopsis

Redirects given channel to a dialplan target

Description

Sends the specified channel to the specified extension priority

This application sets the following channel variables upon completion

- CHANNELREDIRECT_STATUS - Are set to the result of the redirection
 - NOCHANNEL
 - SUCCESS

Syntax

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

Arguments

- channel
- context
- extension
- priority

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Chanspy

ChanSpy()

Synopsis

Listen to a channel, and optionally whisper into it.

Description

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

While spying, the following actions may be performed:

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



Note

The `X` option supersedes the three features above in that if a valid single digit extension exists in the correct context `ChanSpy` will exit to it. This also disables choosing a channel based on `chanprefix` and a digit sequence.

Syntax

```
ChanSpy(chanprefix,options)
```

Arguments

- `chanprefix`
- `options`
 - `b` - Only spy on channels involved in a bridged call.
 - `B` - Instead of whispering on a single channel barge in on both channels involved in the call.
 - `c`
 - `digit` - Specify a DTMF digit that can be used to spy on the next available channel.
 - `d` - Override the typical numeric DTMF functionality and instead use DTMF to switch between spy modes.
 - `4` - spy mode
 - `5` - whisper mode
 - `6` - barge mode
 - `e` - Enable **enforced** mode, so the spying channel can only monitor extensions whose name is in the `ext` : delimited list.
 - `ext`
 - `E` - Exit when the spied-on channel hangs up.
 - `g`
 - `grp` - Only spy on channels in which one or more of the groups listed in `grp` matches one or more groups from the `SPYG` ROUP variable set on the channel to be spied upon.
 - `n` - Say the name of the person being spied on if that person has recorded his/her name. If a context is specified, then that voicemail context will be searched when retrieving the name, otherwise the `default` context be used when searching for the name (i.e. if `SIP/1000` is the channel being spied on and no mailbox is specified, then `1000` will be used when searching for the name).
 - `mailbox`
 - `context`
 - `o` - Only listen to audio coming from this channel.
 - `q` - Don't play a beep when beginning to spy on a channel, or speak the selected channel name.
 - `r` - Record the session to the monitor spool directory. An optional base for the filename may be specified. The default is `chanspy`.
 - `basename`

- `s` - Skip the playback of the channel type (i.e. SIP, IAX, etc) when speaking the selected channel name.
- `S` - Stop when no more channels are left to spy on.
- `v` - Adjust the initial volume in the range from -4 to 4. A negative value refers to a quieter setting.
 - `value`
- `w` - Enable `whisper` mode, so the spying channel can talk to the spied-on channel.
- `W` - Enable `private whisper` mode, so the spying channel can talk to the spied-on channel but cannot listen to that channel.
- `x`
 - `digit` - Specify a DTMF digit that can be used to exit the application.
- `X` - Allow the user to exit ChanSpy to a valid single digit numeric extension in the current context or the context specified by the `SPY_EXIT_CONTEXT` channel variable. The name of the last channel that was spied on will be stored in the `SPY_CHANNEL` variable.

See Also

- [Asterisk 12 Application_ExtenSpy](#)
- [Asterisk 12 ManagerEvent_ChanspyStart](#)
- [Asterisk 12 ManagerEvent_ChanspyStop](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_ConfBridge

ConfBridge()

Synopsis

Conference bridge application.

Description

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

Syntax

```
ConfBridge(conference,bridge_profile,user_profile,menu)
```

Arguments

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

See Also

- [Asterisk 12 Application_ConfBridge](#)
- [Asterisk 12 Function_CONFBRIDGE](#)
- [Asterisk 12 Function_CONFBRIDGE_INFO](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_Busy](#)
- [Asterisk 12 Application_Progress](#)
- [Asterisk 12 Application_Playtones](#)
- [Asterisk 12 Application_Hangup](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_While](#)
- [Asterisk 12 Application_EndWhile](#)
- [Asterisk 12 Application_ExitWhile](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ControlPlayback

ControlPlayback()

Synopsis

Play a file with fast forward and rewind.

Description

This application will play back the given *filename*.

It sets the following channel variables upon completion:

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

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_DAHDI Barge

DAHDI Barge()

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

```
DAHDI Barge (channel )
```

Arguments

- `channel` - Channel to barge.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_DAHDIRAS

DAHDIRAS()

Synopsis

Executes DAHDI ISDN RAS application.

Description

Executes a RAS server using pppd on the given channel. The channel must be a clear channel (i.e. PRI source) and a DAHDI channel to be able to use this function (No modem emulation is included).

Your pppd must be patched to be DAHDI aware.

Syntax

```
DAHDIRAS ( args )
```

Arguments

- `args` - A list of parameters to pass to the pppd daemon, separated by `,` characters.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_DAHDIScan

DAHDIScan()

Synopsis

Scan DAHDI channels to monitor calls.

Description

Allows a call center manager to monitor DAHDI channels in a convenient way. Use # to select the next channel and use * to exit.

Syntax

```
DAHDIScan(group)
```

Arguments

- *group* - Limit scanning to a channel *group* by setting this option.

See Also

- [Asterisk 12 ManagerEvent_ChanspyStart](#)
- [Asterisk 12 ManagerEvent_ChanspyStop](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_DAHDISendCallreroutingFacility

DAHDISendCallreroutingFacility()

Synopsis

Send an ISDN call rerouting/deflection facility message.

Description

This application will send an ISDN switch specific call rerouting/deflection facility message over the current channel. Supported switches depend upon the version of libpri in use.

Syntax

```
DAHDISendCallreroutingFacility(destination,original,reason)
```

Arguments

- `destination` - Destination number.
- `original` - Original called number.
- `reason` - Diversion reason, if not specified defaults to `unknown`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_**DAHDISendKeypadFacility**

DAHDISendKeypadFacility()

Synopsis

Send digits out of band over a PRI.

Description

This application will send the given string of digits in a Keypad Facility IE over the current channel.

Syntax

```
DAHDISendKeypadFacility(digits)
```

Arguments

- `digits`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_DateTime

DateTime()

Synopsis

Says a specified time in a custom format.

Description

Say the date and time in a specified format.

Syntax

```
DateTime(unixtime,timezone,format)
```

Arguments

- `unixtime` - time, in seconds since Jan 1, 1970. May be negative. Defaults to now.
- `timezone` - timezone, see `/usr/share/zoneinfo` for a list. Defaults to machine default.
- `format` - a format the time is to be said in. See `voicemail.conf`. Defaults to `ABdY "digits/at" IMp`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_DBdel

DBdel()

Synopsis

Delete a key from the asterisk database.

Description

This application will delete a *key* from the Asterisk database.



Note

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

Syntax

```
DBdel(family/key)
```

Arguments

- *family*
- *key*

See Also

- [Asterisk 12 Function_DB_DELETE](#)
- [Asterisk 12 Application_DBdeltree](#)
- [Asterisk 12 Function_DB](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Function_DB_DELETE](#)
- [Asterisk 12 Application_DBdel](#)
- [Asterisk 12 Function_DB](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 `AGISIGHUP` channel variable to `no` before executing the AGI application. Alternatively, if you would like the AGI application to exit immediately after a channel hangup is detected, set the `AGIEXITONHANGUP` variable to `yes`.

Use the CLI command `agi show commands` to list available agi commands.

This application sets the following channel variable upon completion:

- `AGISTATUS` - The status of the attempt to the run the AGI script text string, one of:
 - `SUCCESS`
 - `FAILURE`
 - `NOTFOUND`
 - `HANGUP`

Syntax

```
DeadAGI(commandarg1arg2[...])
```

Arguments

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

See Also

- [Asterisk 12 Application_AGI](#)
- [Asterisk 12 Application_EAGI](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Dial

Dial()

Synopsis

Attempt to connect to another device or endpoint and bridge the call.

Description

This application will place calls to one or more specified channels. As soon as one of the requested channels answers, the originating channel will be answered, if it has not already been answered. These two channels will then be active in a bridged call. All other channels that were requested will then be hung up.

Unless there is a timeout specified, the Dial application will wait indefinitely until one of the called channels answers, the user hangs up, or if all of the called channels are busy or unavailable. Dialplan executing will continue if no requested channels can be called, or if the timeout expires. This application will report normal termination if the originating channel hangs up, or if the call is bridged and either of the parties in the bridge ends the call.

If the `OUTBOUND_GROUP` variable is set, all peer channels created by this application will be put into that group (as in `Set(GROUP)=...`). If the `OUTBOUND_GROUP_ONCE` variable is set, all peer channels created by this application will be put into that group (as in `Set(GROUP)=...`). Unlike `OUTBOUND_GROUP`, however, the variable will be unset after use.

This application sets the following channel variables:

- `DIALEDTIME` - This is the time from dialing a channel until when it is disconnected.
- `ANSWEREDTIME` - This is the amount of time for actual call.
- `DIALSTATUS` - This is the status of the call
 - `CHANUNAVAIL`
 - `CONGESTION`
 - `NOANSWER`
 - `BUSY`
 - `ANSWER`
 - `CANCEL`
 - `DONTCALL` - For the Privacy and Screening Modes. Will be set if the called party chooses to send the calling party to the 'Go Away' script.
 - `TORTURE` - For the Privacy and Screening Modes. Will be set if the called party chooses to send the calling party to the 'torture' script.
 - `INVALIDARGS`

Syntax

```
Dial(Technology/Resource[&Technology2/Resource2[&...]][,timeout[,options,URL]])
```

Arguments

- `Technology/Resource`
 - `Technology/Resource` - Specification of the device(s) to dial. These must be in the format of `Technology/Resource`, where *Technology* represents a particular channel driver, and *Resource* represents a resource available to that particular channel driver.
 - `Technology2/Resource2` - Optional extra devices to dial in parallel
If you need more than 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.
 - `b` - Before initiating an outgoing call, Gosub to the specified location using the newly created channel. The Gosub will be

executed for each destination channel.

- context
- exten
- priority
 - arg1
 - argN
- B - Before initiating the outgoing call(s), Gosub to the specified location using the current channel.
 - context
 - exten
 - priority
 - arg1
 - argN
- C - Reset the call detail record (CDR) for this call.
- c - If the Dial() application cancels this call, always set HANGUPCAUSE to 'answered elsewhere'
- d - Allow the calling user to dial a 1 digit extension while waiting for a call to be answered. Exit to that extension if it exists in the current context, or the context defined in the EXITCONTEXT variable, if it exists.
- D - Send the specified DTMF strings **after** the called party has answered, but before the call gets bridged. The *called* DTMF string is sent to the called party, and the *calling* DTMF string is sent to the calling party. Both arguments can be used alone. If *progress* is specified, its DTMF is sent to the called party immediately after receiving a PROGRESS message. See SendDTMF for valid digits.
 - called
 - calling
 - progress
- e - Execute the *h* extension for peer after the call ends
- f - If *x* is not provided, force the CallerID sent on a call-forward or deflection to the dialplan extension of this Dial() using a dialplan *hint*. For example, some PSTNs do not allow CallerID to be set to anything other than the numbers assigned to you. If *x* is provided, force the CallerID sent to *x*.
 - x
- F - When the caller hangs up, transfer the **called** party to the specified destination and **start** execution at that location.
 - context
 - exten
 - priority
- F - When the caller hangs up, transfer the **called** party to the next priority of the current extension and **start** execution at that location.
- g - Proceed with dialplan execution at the next priority in the current extension if the destination channel hangs up.
- G - If the call is answered, transfer the calling party to the specified *priority* and the called party to the specified *priority* plus one.
 - context
 - exten
 - priority
- h - Allow the called party to hang up by sending the DTMF sequence defined for disconnect in *features.conf*.
- H - Allow the calling party to hang up by sending the DTMF sequence defined for disconnect in *features.conf*.
- i - Asterisk will ignore any forwarding requests it may receive on this dial attempt.
- I - Asterisk will ignore any connected line update requests or any redirecting party update requests it may receive on this dial attempt.
- k - Allow the called party to enable parking of the call by sending the DTMF sequence defined for call parking in *features.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:

 - LIMIT_PLAYAUDIO_CALLER - If set, this variable causes Asterisk to play the prompts to the caller.
 - YES default: (true)
 - NO
 - LIMIT_PLAYAUDIO_CALLEE - If set, this variable causes Asterisk to play the prompts to the callee.
 - YES
 - NO default: (true)
 - LIMIT_TIMEOUT_FILE - If specified, *filename* specifies the sound prompt to play when the timeout is reached. If not set, the time remaining will be announced.

- **FILENAME**
- **LIMIT_CONNECT_FILE** - If specified, *filename* specifies the sound prompt to play when the call begins. If not set, the time remaining will be announced.
 - **FILENAME**
- **LIMIT_WARNING_FILE** - If specified, *filename* specifies the sound prompt to play as a warning when time *x* is reached. If not set, the time remaining will be announced.
 - **FILENAME**
- **x** - Maximum call time, in milliseconds
- **y** - Warning time, in milliseconds
- **z** - Repeat time, in milliseconds
- **m** - Provide hold music to the calling party until a requested channel answers. A specific music on hold *class* (as defined in `musiconhold.conf`) can be specified.
 - **class**
- **M** - Execute the specified *macro* for the **called** channel before connecting to the calling channel. Arguments can be specified to the Macro using `^` as a delimiter. The macro can set the variable `MACRO_RESULT` to specify the following actions after the macro is finished executing:
 - **MACRO_RESULT** - If set, this action will be taken after the macro finished executing.
 - **ABORT** - Hangup both legs of the call
 - **CONGESTION** - Behave as if line congestion was encountered
 - **BUSY** - Behave as if a busy signal was encountered
 - **CONTINUE** - Hangup the called party and allow the calling party to continue dialplan execution at the next priority
 - **GOTO:[[<CONTEXT>^]<EXTEN>^]<PRIORITY>** - Transfer the call to the specified destination.
 - **macro** - Name of the macro that should be executed.
 - **arg** - Macro arguments
- **n** - This option is a modifier for the call screening/privacy mode. (See the **p** and **P** options.) It specifies that no introductions are to be saved in the `priv-callerintros` directory.
 - **delete** - With *delete* either not specified or set to 0, the recorded introduction will not be deleted if the caller hangs up while the remote party has not yet answered.
With *delete* set to 1, the introduction will always be deleted.
- **N** - This option is a modifier for the call screening/privacy mode. It specifies that if `Caller*ID` is present, do not screen the call.
- **o** - If *x* is not provided, specify that the `CallerID` that was present on the **calling** channel be stored as the `CallerID` on the **called** channel. This was the behavior of Asterisk 1.0 and earlier. If *x* is provided, specify the `CallerID` stored on the **called** channel. Note that `o(${CALLERID(all)})` is similar to option **o** without the parameter.
 - **x**
- **O** - Enables **operator services** mode. This option only works when bridging a DAHDI channel to another DAHDI channel only. if specified on non-DAHDI interfaces, it will be ignored. When the destination answers (presumably an operator services station), the originator no longer has control of their line. They may hang up, but the switch will not release their line until the destination party (the operator) hangs up.
 - **mode** - With *mode* either not specified or set to 1, the originator hanging up will cause the phone to ring back immediately.
With *mode* set to 2, when the operator flashes the trunk, it will ring their phone back.
- **p** - This option enables screening mode. This is basically Privacy mode without memory.
- **P** - Enable privacy mode. Use *x* as the family/key in the `AstDB` database if it is provided. The current extension is used if a database family/key is not specified.
 - **x**
- **r** - Default: Indicate ringing to the calling party, even if the called party isn't actually ringing. Pass no audio to the calling party until the called channel has answered.
 - **tone** - Indicate progress to calling party. Send audio 'tone' from the `indications.conf` tonezone currently in use.
- **S** - Hang up the call *x* seconds **after** the called party has answered the call.
 - **x**
- **s** - Force the outgoing callerid tag parameter to be set to the string *x*.
Works with the **f** option.
 - **x**
- **t** - Allow the called party to transfer the calling party by sending the DTMF sequence defined in `features.conf`. This setting does not perform policy enforcement on transfers initiated by other methods.
- **T** - Allow the calling party to transfer the called party by sending the DTMF sequence defined in `features.conf`. This setting does not perform policy enforcement on transfers initiated by other methods.
- **U** - Execute via Gosub the routine *x* for the **called** channel before connecting to the calling channel. Arguments can be specified

to the Gosub using ^ as a delimiter. The Gosub routine can set the variable `GOSUB_RESULT` to specify the following actions after the Gosub returns.

- `GOSUB_RESULT`
 - `ABORT` - Hangup both legs of the call.
 - `CONGESTION` - Behave as if line congestion was encountered.
 - `BUSY` - Behave as if a busy signal was encountered.
 - `CONTINUE` - Hangup the called party and allow the calling party to continue dialplan execution at the next priority.
 - `GOTO:[(<CONTEXT>^)<EXTEN>^]<PRIORITY>` - Transfer the call to the specified destination.
- `x` - Name of the subroutine to execute via Gosub
- `arg` - Arguments for the Gosub routine
- `u` - Works with the `f` option.
 - `x` - Force the outgoing callerid presentation indicator parameter to be set to one of the values passed in `x`: `allowed_not_screened` `allowed_passed_screen` `allowed_failed_screen` `allowed_prohib_not_screened` `prohib_passed_screen` `prohib_failed_screen` `prohib_unavailable`
- `w` - Allow the called party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in `features.conf`.
- `W` - Allow the calling party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in `features.conf`.
- `x` - Allow the called party to enable recording of the call by sending the DTMF sequence defined for one-touch automixmonitor in `features.conf`.
- `X` - Allow the calling party to enable recording of the call by sending the DTMF sequence defined for one-touch automixmonitor in `features.conf`.
- `z` - On a call forward, cancel any dial timeout which has been set for this call.
- `URL` - The optional URL will be sent to the called party if the channel driver supports it.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402127

Asterisk 12 Application_Dictate

Dictate()

Synopsis

Virtual Dictation Machine.

Description

Start dictation machine using optional *base_dir* for files.

Syntax

```
Dictate(base_dir,filename)
```

Arguments

- `base_dir`
- `filename`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Directory

Directory()

Synopsis

Provide directory of voicemail extensions.

Description

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

This application will immediately exit if one of the following DTMF digits are received and the extension to jump to exists:

0 - Jump to the 'o' extension, if it exists.

- - Jump to the 'a' extension, if it exists.

Syntax

```
Directory(vm-context[,dial-context[,options]])
```

Arguments

- `vm-context` - This is the context within `voicemail.conf` to use for the Directory. If not specified and `searchcontexts=no` in `voicemail.conf`, then default will be assumed.
- `dial-context` - This is the dialplan context to use when looking for an extension that the user has selected, or when jumping to the `o` or `a` extension. If not specified, the current context will be used.
- `options`
 - `e` - In addition to the name, also read the extension number to the caller before presenting dialing options.
 - `f` - Allow the caller to enter the first name of a user in the directory instead of using the last name. If specified, the optional number argument will be used for the number of characters the user should enter.
 - `n`
 - `l` - Allow the caller to enter the last name of a user in the directory. This is the default. If specified, the optional number argument will be used for the number of characters the user should enter.
 - `n`
 - `b` - Allow the caller to enter either the first or the last name of a user in the directory. If specified, the optional number argument will be used for the number of characters the user should enter.
 - `n`
 - `a` - Allow the caller to additionally enter an alias for a user in the directory. This option must be specified in addition to the `f`, `l`, or `b` option.
 - `m` - Instead of reading each name sequentially and asking for confirmation, create a menu of up to 8 names.
 - `n` - Read digits even if the channel is not answered.
 - `p` - Pause for `n` milliseconds after the digits are typed. This is helpful for people with cellphones, who are not holding the receiver to their ear while entering DTMF.
 - `n`



Note

Only one of the `f`, `l`, or `b` options may be specified. **If more than one is specified**, then Directory will act as if `b` was specified. The number of characters for the user to type defaults to 3.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_DISA

DISA()

Synopsis

Direct Inward System Access.

Description

The DISA, Direct Inward System Access, application allows someone from outside the telephone switch (PBX) to obtain an **internal** system dialtone and to place calls from it as if they were placing a call from within the switch. DISA plays a dialtone. The user enters their numeric passcode, followed by the pound sign #. If the passcode is correct, the user is then given system dialtone within *context* on which a call may be placed. If the user enters an invalid extension and extension *i* exists in the specified *context*, it will be used.

Be aware that using this may compromise the security of your PBX.

The arguments to this application (in `extensions.conf`) allow either specification of a single global *passcode* (that everyone uses), or individual passcodes contained in a file (*filename*).

The file that contains the passcodes (if used) allows a complete specification of all of the same arguments available on the command line, with the sole exception of the options. The file may contain blank lines, or comments starting with # or ; .

Syntax

```
DISA(passcode|filename,context,cidmailbox[@context],options)
```

Arguments

- `passcode|filename` - If you need to present a DISA dialtone without entering a password, simply set *passcode* to `no-password`. You may 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 12 Application_Authenticate](#)
- [Asterisk 12 Application_VMAAuthenticate](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_NoOp](#)
- [Asterisk 12 Application_Verbose](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 `AGISIGHUP` channel variable to `no` before executing the AGI application. Alternatively, if you would like the AGI application to exit immediately after a channel hangup is detected, set the `AGIEXITONHANGUP` variable to `yes`.

Use the CLI command `agi show commands` to list available agi commands.

This application sets the following channel variable upon completion:

- `AGISTATUS` - The status of the attempt to the run the AGI script text string, one of:
 - `SUCCESS`
 - `FAILURE`
 - `NOTFOUND`
 - `HANGUP`

Syntax

```
EAGI(commandarg1arg2[...])
```

Arguments

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

See Also

- [Asterisk 12 Application_AGI](#)
- [Asterisk 12 Application_DeadAGI](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Echo

Echo()

Synopsis

Echo media, DTMF back to the calling party

Description

Echos back any media or DTMF frames read from the calling channel back to itself. This will not echo CONTROL, MODEM, or NULL frames. Note: If '#' detected application exits.

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

Syntax

```
Echo( )
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_EndWhile

EndWhile()

Synopsis

End a while loop.

Description

Return to the previous called `while()`.

Syntax

```
EndWhile()
```

Arguments

See Also

- [Asterisk 12 Application_While](#)
- [Asterisk 12 Application_ExitWhile](#)
- [Asterisk 12 Application_ContinueWhile](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_ExecIf

ExecIf()

Synopsis

Executes dialplan application, conditionally.

Description

If *expr* is true, execute and return the result of *appiftrue(args)*.

If *expr* is true, but *appiftrue* is not found, then the application will return a non-zero value.

Syntax

```
ExecIf(expressionappiftrue[:appiffalse])
```

Arguments

- expression
- execapp
 - appiftrue
 - args
 - appiffalse
 - args

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ExecIfTime

ExecIfTime()

Synopsis

Conditional application execution based on the current time.

Description

This application will execute the specified dialplan application, with optional arguments, if the current time matches the given time specification.

Syntax

```
ExecIfTime(timesweekdaysmdaysmonths[timezone]appargs)
```

Arguments

- day_condition
 - times
 - weekdays
 - mdays
 - months
 - timezone
- appname
 - appargs

See Also

- [Asterisk 12 Application_Exec](#)
- [Asterisk 12 Application_ExecIf](#)
- [Asterisk 12 Application_TryExec](#)
- [Asterisk 12 Application_GotoIfTime](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_While](#)
- [Asterisk 12 Application_EndWhile](#)
- [Asterisk 12 Application_ContinueWhile](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ExtenSpy

ExtenSpy()

Synopsis

Listen to a channel, and optionally whisper into it.

Description

This application is used to listen to the audio from an Asterisk channel. This includes the audio coming in and out of the channel being spied on. Only channels created by outgoing calls for the specified extension will be selected for spying. If the optional context is not supplied, the current channel's context will be used.

While spying, the following actions may be performed:

- Dialing # cycles the volume level.
- Dialing * will stop spying and look for another channel to spy on.



Note

The *X* option supersedes the three features above in that if a valid single digit extension exists in the correct context ChanSpy will exit to it. This also disables choosing a channel based on *chanprefix* and a digit sequence.

Syntax

```
ExtenSpy ( exten@context , options )
```

Arguments

- *exten*
 - *exten* - Specify extension.
 - *context* - Optionally specify a context, defaults to *default*.
- *options*
 - *b* - Only spy on channels involved in a bridged call.
 - *B* - Instead of whispering on a single channel barge in on both channels involved in the call.
 - *c*
 - *digit* - Specify a DTMF digit that can be used to spy on the next available channel.
 - *d* - Override the typical numeric DTMF functionality and instead use DTMF to switch between spy modes.
 - 4 - spy mode
 - 5 - whisper mode
 - 6 - barge mode
 - *e* - Enable **enforced** mode, so the spying channel can only monitor extensions whose name is in the *ext* : delimited list.
 - *ext*
 - *E* - Exit when the spied-on channel hangs up.
 - *g*
 - *grp* - Only spy on channels in which one or more of the groups listed in *grp* matches one or more groups from the *SPYG ROUP* variable set on the channel to be spied upon.
 - *n* - Say the name of the person being spied on if that person has recorded his/her name. If a context is specified, then that voicemail context will be searched when retrieving the name, otherwise the *default* context be used when searching for the name (i.e. if SIP/1000 is the channel being spied on and no mailbox is specified, then 1000 will be used when searching for the name).
 - *mailbox*
 - *context*
 - *o* - Only listen to audio coming from this channel.
 - *q* - Don't play a beep when beginning to spy on a channel, or speak the selected channel name.
 - *r* - Record the session to the monitor spool directory. An optional base for the filename may be specified. The default is *chanspy*.
 - *basename*

- `s` - Skip the playback of the channel type (i.e. SIP, IAX, etc) when speaking the selected channel name.
- `S` - Stop when there are no more extensions left to spy on.
- `v` - Adjust the initial volume in the range from -4 to 4. A negative value refers to a quieter setting.
 - `value`
- `w` - Enable `whisper` mode, so the spying channel can talk to the spied-on channel.
- `W` - Enable `private whisper` mode, so the spying channel can talk to the spied-on channel but cannot listen to that channel.
- `x`
 - `digit` - Specify a DTMF digit that can be used to exit the application.
- `X` - Allow the user to exit ChanSpy to a valid single digit numeric extension in the current context or the context specified by the `SPY_EXIT_CONTEXT` channel variable. The name of the last channel that was spied on will be stored in the `SPY_CHANNEL` variable.

See Also

- [Asterisk 12 Application_Chanspy](#)
- [Asterisk 12 ManagerEvent_ChanspyStart](#)
- [Asterisk 12 ManagerEvent_ChanspyStop](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ExternalIVR

ExternalIVR()

Synopsis

Interfaces with an external IVR application.

Description

Either forks a process to run given command or makes a socket to connect to given host and starts a generator on the channel. The generator's play list is controlled by the external application, which can add and clear entries via simple commands issued over its stdout. The external application will receive all DTMF events received on the channel, and notification if the channel is hung up. The received on the channel, and notification if the channel is hung up. The application will not be forcibly terminated when the channel is hung up. For more information see `doc/AST.pdf`.

Syntax

```
ExternalIVR(arg1arg2[...],options)
```

Arguments

- `command|ivr://host`
 - `arg1`
 - `arg2`
- `options`
 - `n` - Tells ExternalIVR() not to answer the channel.
 - `i` - Tells ExternalIVR() not to send a hangup and exit when the channel receives a hangup, instead it sends an `I` informative message meaning that the external application MUST hang up the call with an `H` command.
 - `d` - Tells ExternalIVR() to run on a channel that has been hung up and will not look for hangups. The external application must exit with an `E` command.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 12 Application_SendDTMF](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_FollowMe

FollowMe()

Synopsis

Find-Me/Follow-Me application.

Description

This application performs Find-Me/Follow-Me functionality for the caller as defined in the profile matching the *followmeid* parameter in *followme.conf*. If the specified *followmeid* profile doesn't exist in *followme.conf*, execution will be returned to the dialplan and call execution will continue at the next priority.

Returns -1 on hangup.

Syntax

```
FollowMe(followmeid,options)
```

Arguments

- *followmeid*
- *options*
 - *a* - Record the caller's name so it can be announced to the callee on each step.
 - *B* - Before initiating the outgoing call(s), Gosub to the specified location using the current channel.
 - *context*
 - *exten*
 - *priority*
 - *arg1*
 - *argN*
 - *b* - Before initiating an outgoing call, Gosub to the specified location using the newly created channel. The Gosub will be executed for each destination channel.
 - *context*
 - *exten*
 - *priority*
 - *arg1*
 - *argN*
 - *d* - Disable the 'Please hold while we try to connect your call' announcement.
 - *I* - Asterisk will ignore any connected line update requests it may receive on this dial attempt.
 - *l* - Disable local call optimization so that applications with audio hooks between the local bridge don't get dropped when the calls get joined directly.
 - *N* - Don't answer the incoming call until we're ready to connect the caller or give up.
 - *n* - Playback the unreachable status message if we've run out of steps or the callee has elected not to be reachable.
 - *s* - Playback the incoming status message prior to starting the follow-me step(s)

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ForkCDR

ForkCDR()

Synopsis

Forks the current Call Data Record for this channel.

Description

Causes the Call Data Record engine to fork a new CDR starting from the time the application is executed. The forked CDR will be linked to the end of the CDRs associated with the channel.

Syntax

`ForkCDR(options)`

Arguments

- `options`
 - `a` - If the channel is answered, set the answer time on the forked CDR to the current time. If this option is not used, the answer time on the forked CDR will be the answer time on the original CDR. If the channel is not answered, this option has no effect. Note that this option is implicitly assumed if the `x` option is used.
 - `e` - End (finalize) the original CDR.
 - `x` - Reset the start and answer times on the forked CDR. This will set the start and answer times (if the channel is answered) to be set to the current time. Note that this option implicitly assumes the `a` option.
 - `v` - Do not copy CDR variables and attributes from the original CDR to the forked CDR.

See Also

- [Asterisk 12 Function_CDR](#)
- [Asterisk 12 Application_NoCDR](#)
- [Asterisk 12 Application_ResetCDR](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_GetCPEID

GetCPEID()

Synopsis

Get ADSI CPE ID.

Description

Obtains and displays ADSI CPE ID and other information in order to properly setup `dahdi.conf` for on-hook operations.

Syntax

```
GetCPEID()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Gosub

Gosub()

Synopsis

Jump to label, saving return address.

Description

Jumps to the label specified, saving the return address.

Syntax

```
Gosub(context,extenarg1[...argN])
```

Arguments

- context
- exten
- priority
 - arg1
 - argN

See Also

- [Asterisk 12 Application_GosubIf](#)
- [Asterisk 12 Application_Macro](#)
- [Asterisk 12 Application_Goto](#)
- [Asterisk 12 Application_Return](#)
- [Asterisk 12 Application_StackPop](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Gosublf

Gosublf()

Synopsis

Conditionally jump to label, saving return address.

Description

If the condition is true, then jump to *labeliftrue*. If false, jumps to *labeliffalse*, if specified. In either case, a jump saves the return point in the dialplan, to be returned to with a *Return*.

Syntax

```
GosubIf(conditionlabeliftrue:labeliffalse)
```

Arguments

- *condition*
- *destination*
 - *labeliftrue* - Continue at *labeliftrue* if the condition is true. Takes the form similar to *Goto()* of *[[context,]extension,]priority*.
 - *arg1*
 - *argN*
 - *labeliffalse* - Continue at *labeliffalse* if the condition is false. Takes the form similar to *Goto()* of *[[context,]extension,]priority*.
 - *arg1*
 - *argN*

See Also

- [Asterisk 12 Application_Gosub](#)
- [Asterisk 12 Application_Return](#)
- [Asterisk 12 Application_MacroIf](#)
- [Asterisk 12 Function_IF](#)
- [Asterisk 12 Application_Gotof](#)
- [Asterisk 12 Application_Goto](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Goto

Goto()

Synopsis

Jump to a particular priority, extension, or context.

Description

This application will set the current context, extension, and priority in the channel structure. After it completes, the pbx engine will continue dialplan execution at the specified location. If no specific *extension*, or *extension* and *context*, are specified, then this application will just set the specified *priority* of the current extension.

At least a *priority* is required as an argument, or the goto will return a -1, and the channel and call will be terminated.

If the location that is put into the channel information is bogus, and asterisk cannot find that location in the dialplan, then the execution engine will try to find and execute the code in the *i* (invalid) extension in the current context. If that does not exist, it will try to execute the *h* extension. If neither the *h* nor *i* 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 12 Application_Gotof](#)
- [Asterisk 12 Application_GotofTime](#)
- [Asterisk 12 Application_Gosub](#)
- [Asterisk 12 Application_Macro](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Gotolf

Gotolf()

Synopsis

Conditional goto.

Description

This application will set the current context, extension, and priority in the channel structure based on the evaluation of the given condition. After this application completes, the pbx engine will continue dialplan execution at the specified location in the dialplan. The labels are specified with the same syntax as used within the Goto application. If the label chosen by the condition is omitted, no jump is performed, and the execution passes to the next instruction. If the target location is bogus, and does not exist, the execution engine will try to find and execute the code in the *i* (invalid) extension in the current context. If that does not exist, it will try to execute the *h* extension. If neither the *h* nor *i* extensions have been defined, the channel is hung up, and the execution of instructions on the channel is terminated. Remember that this command can set the current context, and if the context specified does not exist, then it will not be able to find any 'h' or 'i' extensions there, and the channel and call will both be terminated!.

Syntax

```
GotoIf(conditionlabeliftrue:labeliffalse)
```

Arguments

- *condition*
- *destination*
 - *labeliftrue* - Continue at *labeliftrue* if the condition is true. Takes the form similar to Goto() of [[context,]extension,]priority.
 - *labeliffalse* - Continue at *labeliffalse* if the condition is false. Takes the form similar to Goto() of [[context,]extension,]priority.

See Also

- [Asterisk 12 Application_Goto](#)
- [Asterisk 12 Application_GotolfTime](#)
- [Asterisk 12 Application_Gosublf](#)
- [Asterisk 12 Application_Macrof](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_GotolfTime

GotolfTime()

Synopsis

Conditional Goto based on the current time.

Description

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

Syntax

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

Arguments

- condition
 - times
 - weekdays
 - mdays
 - months
 - timezone
- destination
 - *labeliftrue* - Continue at *labeliftrue* if the condition is true. Takes the form similar to `Goto()` of `[[context,]extension,]priority`.
 - *labeliffalse* - Continue at *labeliffalse* if the condition is false. Takes the form similar to `Goto()` of `[[context,]extension,]priority`.

See Also

- [Asterisk 12 Application_Gotolf](#)
- [Asterisk 12 Application_Goto](#)
- [Asterisk 12 Function_IFTIME](#)
- [Asterisk 12 Function_TESTTIME](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_Answer](#)
- [Asterisk 12 Application_Busy](#)
- [Asterisk 12 Application_Congestion](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_HangupCauseClear

HangupCauseClear()

Synopsis

Clears hangup cause information from the channel that is available through HANGUPCAUSE.

Description

Clears all channel-specific hangup cause information from the channel. This is never done automatically (i.e. for new Dial()s).

Syntax

See Also

- [Asterisk 12 Function_HANGUPCAUSE](#)
- [Asterisk 12 Function_HANGUPCAUSE_KEYS](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_ICES

ICES()

Synopsis

Encode and stream using 'ices'.

Description

Streams to an icecast server using ices (available separately). A configuration file must be supplied for ices (see contrib/asterisk-ices.xml).



Note

ICES version 2 client and server required.

Syntax

```
ICES(config)
```

Arguments

- `config` - ICES configuration file.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ImportVar

ImportVar()

Synopsis

Import a variable from a channel into a new variable.

Description

This application imports a *variable* from the specified *channel* (as opposed to the current one) and stores it as a variable (*newvar*) in the current channel (the channel that is calling this application). Variables created by this application have the same inheritance properties as those created with the `Set` application.

Syntax

```
ImportVar(newvarchannelnamevariable)
```

Arguments

- `newvar`
- `vardata`
 - `channelname`
 - `variable`

See Also

- [Asterisk 12 Application_Set](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Incomplete

Incomplete()

Synopsis

Returns AST_PBX_INCOMPLETE value.

Description

Signals the PBX routines that the previous matched extension is incomplete and that further input should be allowed before matching can be considered to be complete. Can be used within a pattern match when certain criteria warrants a longer match.

Syntax

```
Incomplete(n)
```

Arguments

- `n` - If specified, then Incomplete will not attempt to answer the channel first.



Note

Most channel types need to be in Answer state in order to receive DTMF.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_IVRDemo

IVRDemo()

Synopsis

IVR Demo Application.

Description

This is a skeleton application that shows you the basic structure to create your own asterisk applications and demonstrates the IVR demo.

Syntax

```
IVRDemo(filename)
```

Arguments

- filename

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberJoin_res_jabber

JabberJoin() - [res_jabber]

Synopsis

Join a chat room

Description

Allows Asterisk to join a chat room.

Syntax

```
JabberJoin(Jabber,RoomJID[,Nickname])
```

Arguments

- `Jabber` - Client or transport Asterisk uses to connect to Jabber.
- `RoomJID` - XMPP/Jabber JID (Name) of chat room.
- `Nickname` - The nickname Asterisk will use in the chat room.



Note

If a different nickname is supplied to an already joined room, the old nick will be changed to the new one.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberJoin_res_xmpp

JabberJoin() - [res_xmpp]

Synopsis

Join a chat room

Description

Allows Asterisk to join a chat room.

Syntax

```
JabberJoin(Jabber,RoomJID[,Nickname])
```

Arguments

- **Jabber** - Client or transport Asterisk uses to connect to Jabber.
- **RoomJID** - XMPP/Jabber JID (Name) of chat room.
- **Nickname** - The nickname Asterisk will use in the chat room.



Note

If a different nickname is supplied to an already joined room, the old nick will be changed to the new one.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberLeave_res_jabber

JabberLeave() - [res_jabber]

Synopsis

Leave a chat room

Description

Allows Asterisk to leave a chat room.

Syntax

```
JabberLeave(Jabber,RoomJID[,Nickname])
```

Arguments

- `Jabber` - Client or transport Asterisk uses to connect to Jabber.
- `RoomJID` - XMPP/Jabber JID (Name) of chat room.
- `Nickname` - The nickname Asterisk uses in the chat room.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberLeave_res_xmpp

JabberLeave() - [res_xmpp]

Synopsis

Leave a chat room

Description

Allows Asterisk to leave a chat room.

Syntax

```
JabberLeave(Jabber,RoomJID[,Nickname])
```

Arguments

- `Jabber` - Client or transport Asterisk uses to connect to Jabber.
- `RoomJID` - XMPP/Jabber JID (Name) of chat room.
- `Nickname` - The nickname Asterisk uses in the chat room.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberSend_res_jabber

JabberSend() - [res_jabber]

Synopsis

Sends an XMPP message to a buddy.

Description

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

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

Syntax

```
JabberSend(account,jid,message)
```

Arguments

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

See Also

- [Asterisk 12 Function_JABBER_STATUS_res_jabber](#)
- [Asterisk 12 Function_JABBER_RECEIVE_res_jabber](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberSend_res_xmpp

JabberSend() - [res_xmpp]

Synopsis

Sends an XMPP message to a buddy.

Description

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

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

Syntax

```
JabberSend(account,jid,message)
```

Arguments

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

See Also

- [Asterisk 12 Function_JABBER_STATUS_res_xmpp](#)
- [Asterisk 12 Function_JABBER_RECEIVE_res_xmpp](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberSendGroup_res_jabber

JabberSendGroup() - [res_jabber]

Synopsis

Send a Jabber Message to a specified chat room

Description

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



Note

To be able to send messages to a chat room, a user must have previously joined it. Use the *JabberJoin* function to do so.

Syntax

```
JabberSendGroup(Jabber,RoomJID,Message[ ,Nickname])
```

Arguments

- **Jabber** - Client or transport Asterisk uses to connect to Jabber.
- **RoomJID** - XMPP/Jabber JID (Name) of chat room.
- **Message** - Message to be sent to the chat room.
- **Nickname** - The nickname Asterisk uses in the chat room.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberSendGroup_res_xmpp

JabberSendGroup() - [res_xmpp]

Synopsis

Send a Jabber Message to a specified chat room

Description

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



Note

To be able to send messages to a chat room, a user must have previously joined it. Use the *JabberJoin* function to do so.

Syntax

```
JabberSendGroup(Jabber,RoomJID,Message[ ,Nickname])
```

Arguments

- **Jabber** - Client or transport Asterisk uses to connect to Jabber.
- **RoomJID** - XMPP/Jabber JID (Name) of chat room.
- **Message** - Message to be sent to the chat room.
- **Nickname** - The nickname Asterisk uses in the chat room.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberStatus_res_jabber

JabberStatus() - [res_jabber]

Synopsis

Retrieve the status of a jabber list member

Description

This application is deprecated. Please use the JABBER_STATUS() function instead.

Retrieves the numeric status associated with the specified buddy *JID*. The return value in the *_Variable_* will be one of the following.

- 1 - Online.
- 2 - Chatty.
- 3 - Away.
- 4 - Extended Away.
- 5 - Do Not Disturb.
- 6 - Offline.
- 7 - Not In Roster.

Syntax

```
JabberStatus(Jabber,JID,Variable)
```

Arguments

- *Jabber* - Client or transport Asterisk users to connect to Jabber.
- *JID* - XMPP/Jabber JID (Name) of recipient.
- *Variable* - Variable to store the status of requested user.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_JabberStatus_res_xmpp

JabberStatus() - [res_xmpp]

Synopsis

Retrieve the status of a jabber list member

Description

This application is deprecated. Please use the JABBER_STATUS() function instead.

Retrieves the numeric status associated with the specified buddy *JID*. The return value in the *_Variable_* will be one of the following.

- 1 - Online.
- 2 - Chatty.
- 3 - Away.
- 4 - Extended Away.
- 5 - Do Not Disturb.
- 6 - Offline.
- 7 - Not In Roster.

Syntax

```
JabberStatus(Jabber,JID,Variable)
```

Arguments

- *Jabber* - Client or transport Asterisk users to connect to Jabber.
- *JID* - XMPP/Jabber JID (Name) of recipient.
- *Variable* - Variable to store the status of requested user.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_Macro

Macro()

Synopsis

Macro Implementation.

Description

Executes a macro using the context macro- *name*, jumping to the *s* extension of that context and executing each step, then returning when the steps end.

The calling extension, context, and priority are stored in `MACRO_EXTEN`, `MACRO_CONTEXT` and `MACRO_PRIORITY` respectively. Arguments become `ARG1`, `ARG2`, etc in the macro context.

If you Goto out of the Macro context, the Macro will terminate and control will be returned at the location of the Goto.

If `MACRO_OFFSET` is set at termination, Macro will attempt to continue at priority `MACRO_OFFSET + N + 1` if such a step exists, and `N + 1` otherwise.



Warning

Because of the way Macro is implemented (it executes the priorities contained within it via sub-engine), and a fixed per-thread memory stack allowance, macros are limited to 7 levels of nesting (macro calling macro calling macro, etc.); It may be possible that stack-intensive applications in deeply nested macros could cause asterisk to crash earlier than this limit. It is advised that if you need to deeply nest macro calls, that you use the Gosub application (now allows arguments like a Macro) with explicit `Return()` calls instead.



Warning

Use of the application `WaitExten` within a macro will not function as expected. Please use the `Read` application in order to read DTMF from a channel currently executing a macro.

Syntax

```
Macro(namearg1arg2[...])
```

Arguments

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

See Also

- [Asterisk 12 Application_MacroExit](#)
- [Asterisk 12 Application_Goto](#)
- [Asterisk 12 Application_Gosub](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MacroExclusive

MacroExclusive()

Synopsis

Exclusive Macro Implementation.

Description

Executes macro defined in the context macro- *name*. Only one call at a time may run the macro. (we'll wait if another call is busy executing in the Macro)

Arguments and return values as in application Macro()



Warning

Use of the application `WaitExten` within a macro will not function as expected. Please use the `Read` application in order to read DTMF from a channel currently executing a macro.

Syntax

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

Arguments

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

See Also

- [Asterisk 12 Application_Macro](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MacroExit

MacroExit()

Synopsis

Exit from Macro.

Description

Causes the currently running macro to exit as if it had ended normally by running out of priorities to execute. If used outside a macro, will likely cause unexpected behavior.

Syntax

```
MacroExit()
```

Arguments

See Also

- [Asterisk 12 Application_Macro](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MacroIf

MacroIf()

Synopsis

Conditional Macro implementation.

Description

Executes macro defined in *macroiftrue* if *expr* is true (otherwise *macroiffalse* if provided)

Arguments and return values as in application Macro()



Warning

Use of the application `WaitExten` within a macro will not function as expected. Please use the `Read` application in order to read DTMF from a channel currently executing a macro.

Syntax

```
MacroIf(exprmacroiftrue:macroiffalse)
```

Arguments

- `expr`
- `destination`
 - `macroiftrue`
 - `macroiftrue`
 - `arg1`
 - `macroiffalse`
 - `macroiffalse`
 - `arg1`

See Also

- [Asterisk 12 Application_Gotof](#)
- [Asterisk 12 Application_Gosublf](#)
- [Asterisk 12 Function_IF](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MailboxExists

MailboxExists()

Synopsis

Check to see if Voicemail mailbox exists.

Description



Note

DEPRECATED. Use `VM_INFO(mailbox[@context],exists)` instead.

Check to see if the specified *mailbox* exists. If no voicemail *context* is specified, the `default` context will be used.

This application will set the following channel variable upon completion:

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

Syntax

```
MailboxExists(mailbox@context,options)
```

Arguments

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

See Also

- [Asterisk 12 Function_VM_INFO](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MeetMe

MeetMe()

Synopsis

MeetMe conference bridge.

Description

Enters the user into a specified MeetMe conference. If the *confno* is omitted, the user will be prompted to enter one. User can exit the conference by hangup, or if the *p* option is specified, by pressing #.



Note

The DAHDI kernel modules and a functional DAHDI timing source (see *dahdi_test*) must be present for conferencing to operate properly. In addition, the *chan_dahdi* channel driver must be loaded for the *i* and *r* options to operate at all.

Syntax

```
MeetMe(confno,options,pin)
```

Arguments

- *confno* - The conference number
- *options*
 - *a* - Set admin mode.
 - *A* - Set marked mode.
 - *b* - Run AGI script specified in *MEETME_AGI_BACKGROUND* Default: *conf-background.agi*.
 - *c* - Announce user(s) count on joining a conference.
 - *C* - Continue in dialplan when kicked out of conference.
 - *d* - Dynamically add conference.
 - *D* - Dynamically add conference, prompting for a PIN.
 - *e* - Select an empty conference.
 - *E* - Select an empty pinless conference.
 - *F* - Pass DTMF through the conference.
 - *G* - Play an intro announcement in conference.
 - *x* - The file to playback
 - *i* - Announce user join/leave with review.
 - *I* - Announce user join/leave without review.
 - *k* - Close the conference if there's only one active participant left at exit.
 - *l* - Set listen only mode (Listen only, no talking).
 - *m* - Set initially muted.
 - *M* - Enable music on hold when the conference has a single caller. Optionally, specify a *musiconhold* class to use. If one is not provided, it will use the channel's currently set music class, or *default*.
 - *class*
 - *n* - Disable the denoiser. By default, if *func_speex* is loaded, Asterisk will apply a denoiser to channels in the MeetMe conference. However, channel drivers that present audio with a varying rate will experience degraded performance with a denoiser attached. This parameter allows a channel joining the conference to choose not to have a denoiser attached without having to unload *func_speex*.
 - *o* - Set talker optimization - treats talkers who aren't speaking as being muted, meaning (a) No encode is done on transmission and (b) Received audio that is not registered as talking is omitted causing no buildup in background noise.
 - *p* - Allow user to exit the conference by pressing # (default) or any of the defined keys. Dial plan execution will continue at the next priority following MeetMe. The key used is set to channel variable *MEETME_EXIT_KEY*.
 - *keys*
 - *P* - Always prompt for the pin even if it is specified.
 - *q* - Quiet mode (don't play enter/leave sounds).
 - *r* - Record conference (records as *MEETME_RECORDINGFILE* using format *MEETME_RECORDINGFORMAT*. Default filename is *meetme-conf-rec-\${CONFNO}-\${UNIQUEID}* and the default format is *wav*.

- `s` - Present menu (user or admin) when `*` is received (send to menu).
- `t` - Set talk only mode. (Talk only, no listening).
- `T` - Set talker detection (sent to manager interface and meetme list).
- `v` - Announce when a user is joining or leaving the conference. Use the voicemail greeting as the announcement. If the `i` or `I` options are set, the application will fall back to them if no voicemail greeting can be found.
 - `mailbox@context` - The mailbox and voicemail context to play from. If no context provided, assumed context is default.
- `w` - Wait until the marked user enters the conference.
 - `secs`
- `x` - Leave the conference when the last marked user leaves.
- `X` - Allow user to exit the conference by entering a valid single digit extension `MEETME_EXIT_CONTEXT` or the current context if that variable is not defined.
- `1` - Do not play message when first person enters
- `S` - Kick the user `x` seconds **after** he entered into the conference.
 - `x`
- `L` - Limit the conference to `x` ms. Play a warning when `y` ms are left. Repeat the warning every `z` ms. The following special variables can be used with this option:
 - `CONF_LIMIT_TIMEOUT_FILE` - File to play when time is up.
 - `CONF_LIMIT_WARNING_FILE` - File to play as warning if `y` is defined. The default is to say the time remaining.
 - `x`
 - `y`
 - `z`
- `pin`

See Also

- [Asterisk 12 Application_MeetMeCount](#)
- [Asterisk 12 Application_MeetMeAdmin](#)
- [Asterisk 12 Application_MeetMeChannelAdmin](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MeetMeAdmin

MeetMeAdmin()

Synopsis

MeetMe conference administration.

Description

Run admin *command* for conference *confno*.

Will additionally set the variable MEETMEADMINSTATUS with one of the following values:

- MEETMEADMINSTATUS
 - NOPARSE - Invalid arguments.
 - NOTFOUND - User specified was not found.
 - FAILED - Another failure occurred.
 - OK - The operation was completed successfully.

Syntax

```
MeetMeAdmin(confno,command,user)
```

Arguments

- confno
- command
 - e - Eject last user that joined.
 - E - Extend conference end time, if scheduled.
 - k - Kick one user out of conference.
 - K - Kick all users out of conference.
 - l - Unlock conference.
 - L - Lock conference.
 - m - Unmute one user.
 - M - Mute one user.
 - n - Unmute all users in the conference.
 - N - Mute all non-admin users in the conference.
 - r - Reset one user's volume settings.
 - R - Reset all users volume settings.
 - s - Lower entire conference speaking volume.
 - S - Raise entire conference speaking volume.
 - t - Lower one user's talk volume.
 - T - Raise one user's talk volume.
 - u - Lower one user's listen volume.
 - U - Raise one user's listen volume.
 - v - Lower entire conference listening volume.
 - V - Raise entire conference listening volume.
- user

See Also

- [Asterisk 12 Application_MeetMe](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 12 Application_MeetMe](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MessageSend

MessageSend()

Synopsis

Send a text message.

Description

Send a text message. The body of the message that will be sent is what is currently set to `MESSAGE(body)`. The technology chosen for sending the message is determined based on a prefix to the `to` parameter.

This application sets the following channel variables:

- `MESSAGE_SEND_STATUS` - This is the message delivery status returned by this application.
 - `INVALID_PROTOCOL` - No handler for the technology part of the URI was found.
 - `INVALID_URI` - The protocol handler reported that the URI was not valid.
 - `SUCCESS` - Successfully passed on to the protocol handler, but delivery has not necessarily been guaranteed.
 - `FAILURE` - The protocol handler reported that it was unable to deliver the message for some reason.

Syntax

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

Arguments

- `to` - A To URI for the message.

Technology: PJSIP

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

Technology: SIP

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

Technology: XMPP

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

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

Technology: PJSIP

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

Technology: SIP

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

Technology: XMPP

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403796

Asterisk 12 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 Unknown

Asterisk 12 Application_MinivmAccMess

MinivmAccMess()

Synopsis

Record account specific messages.

Description

This application is part of the Mini-Voicemail system, configured in `minivm.conf`.

Use this application to record account specific audio/video messages for busy, unavailable and temporary messages.

Account specific directories will be created if they do not exist.

- `MVM_ACCMESS_STATUS` - This is the result of the attempt to record the specified greeting.
FAILED is set if the file can't be created.
 - SUCCESS
 - FAILED

Syntax

```
MinivmAccMess(username@domain[,options])
```

Arguments

- mailbox
 - username - Voicemail username
 - domain - Voicemail domain
- options
 - u - Record the unavailable greeting.
 - b - Record the busy greeting.
 - t - Record the temporary greeting.
 - n - Account name.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 chosen, but will be overridden if a temporary message exists for the account.

- MVM_GREET_STATUS - This is the status of the greeting playback.
 - SUCCESS
 - USEREXIT
 - FAILED

Syntax

```
MinivmGreet(username@domain[,options])
```

Arguments

- mailbox
 - username - Voicemail username
 - domain - Voicemail domain
- options
 - b - Play the busy greeting to the calling party.
 - s - Skip the playback of instructions for leaving a message to the calling party.
 - u - Play the unavailable greeting.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MinivmMWI

MinivmMWI()

Synopsis

Send Message Waiting Notification to subscriber(s) of mailbox.

Description

This application is part of the Mini-Voicemail system, configured in `minivm.conf`.

MinivmMWI is used to send message waiting indication to any devices whose channels have subscribed to the mailbox passed in the first parameter.

Syntax

```
MinivmMWI(username@domain,urgent,new,old)
```

Arguments

- mailbox
 - username - Voicemail username
 - domain - Voicemail domain
- urgent - Number of urgent messages in mailbox.
- new - Number of new messages in mailbox.
- old - Number of old messages in mailbox.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MinivmNotify

MinivmNotify()

Synopsis

Notify voicemail owner about new messages.

Description

This application is part of the Mini-Voicemail system, configured in minivm.conf.

MiniVMnotify forwards messages about new voicemail to e-mail and pager. If there's no user account for that address, a temporary account will be used with default options (set in minivm.conf).

If the channel variable MVM_COUNTER is set, this will be used in the message file name and available in the template for the message.

If no template is given, the default email template will be used to send email and default pager template to send paging message (if the user account is configured with a paging address).

- MVM_NOTIFY_STATUS - This is the status of the notification attempt
 - SUCCESS
 - FAILED

Syntax

```
MinivmNotify(username@domain[,options])
```

Arguments

- mailbox
 - username - Voicemail username
 - domain - Voicemail domain
- options
 - template - E-mail template to use for voicemail notification

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MinivmRecord

MinivmRecord()

Synopsis

Receive Mini-Voicemail and forward via e-mail.

Description

This application is part of the Mini-Voicemail system, configured in `minivm.conf`

MiniVM records audio file in configured format and forwards message to e-mail and pager.

If there's no user account for that address, a temporary account will be used with default options.

The recorded file name and path will be stored in `MVM_FILENAME` and the duration of the message will be stored in `MVM_DURATION`



Note

If the caller hangs up after the recording, the only way to send the message and clean up is to execute in the `h` extension. The application will exit if any of the following DTMF digits are received and the requested extension exist in the current context.

- `MVM_RECORD_STATUS` - This is the status of the record operation
 - `SUCCESS`
 - `USEREXIT`
 - `FAILED`

Syntax

```
MinivmRecord(username@domain[,options])
```

Arguments

- `mailbox`
 - `username` - Voicemail username
 - `domain` - Voicemail domain
- `options`
 - `0` - Jump to the `o` extension in the current dialplan context.
 - `*` - Jump to the `a` extension in the current dialplan context.
 - `g` - Use the specified amount of gain when recording the voicemail message. The units are whole-number decibels (dB).
 - `gain` - Amount of gain to use

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 preceded by an application such as Answer or Progress().



Note

MixMonitor runs as an audiohook.

- MIXMONITOR_FILENAME - Will contain the filename used to record.

Syntax

```
MixMonitor(filename.extension,options,command)
```

Arguments

- file
 - filename - If *filename* is an absolute path, uses that path, otherwise creates the file in the configured monitoring directory from *asterisk.conf*.
 - extension
- options
 - a - Append to the file instead of overwriting it.
 - b - Only save audio to the file while the channel is bridged.
 - v - Adjust the **heard** volume by a factor of *x* (range -4 to 4)
 - *x*
 - V - Adjust the **spoken** volume by a factor of *x* (range -4 to 4)
 - *x*
 - w - Adjust both, **heard and spoken** volumes by a factor of *x* (range -4 to 4)
 - *x*
 - r - Use the specified file to record the **receive** audio feed. Like with the basic filename argument, if an absolute path isn't given, it will create the file in the configured monitoring directory.
 - file
 - t - Use the specified file to record the **transmit** audio feed. Like with the basic filename argument, if an absolute path isn't given, it will create the file in the configured monitoring directory.
 - file
 - i - Stores the MixMonitor's ID on this channel variable.
 - chanvar
 - m - Create a copy of the recording as a voicemail in the indicated **mailbox(es)** separated by commas eg. m(1111default,...). Folders can be optionally specified using the syntax: mailbox@context/folder
 - mailbox
- command - Will be executed when the recording is over.
Any strings matching `^{\}` will be unescaped to `x`.
All variables will be evaluated at the time MixMonitor is called.

See Also

- [Asterisk 12 Application_Monitor](#)
- [Asterisk 12 Application_StopMixMonitor](#)
- [Asterisk 12 Application_PauseMonitor](#)
- [Asterisk 12 Application_UnpauseMonitor](#)

- [Asterisk 12 Function_AUDIOHOOK_INHERIT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_StopMonitor](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Morsecode

Morsecode()

Synopsis

Plays morse code.

Description

Plays the Morse code equivalent of the passed string.

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

This application uses the following variables:

- `MORSEDTLEN` - 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 12 Application_SayAlpha](#)
- [Asterisk 12 Application_SayPhonetic](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MP3Player

MP3Player()

Synopsis

Play an MP3 file or M3U playlist file or stream.

Description

Executes mpg123 to play the given location, which typically would be a mp3 filename or m3u playlist filename or a URL. Please read <http://en.wikipedia.org/wiki/M3U> to see how M3U playlist file format is like, Example usage would be exten => 1234,1,MP3Player(/var/lib/asterisk/playlist.m3u) User can exit by pressing any key on the dialpad, or by hanging up.

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

Syntax

```
MP3Player(Location)
```

Arguments

- `Location` - Location of the file to be played. (argument passed to mpg123)

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_MSet

MSet()

Synopsis

Set channel variable(s) or function value(s).

Description

This function can be used to set the value of channel variables or dialplan functions. When setting variables, if the variable name is prefixed with `_`, the variable will be inherited into channels created from the current channel. If the variable name is prefixed with `__`, the variable will be inherited into channels created from the current channel and all children channels. MSet behaves in a similar fashion to the way Set worked in 1.2/1.4 and is thus prone to doing things that you may not expect. For example, it strips surrounding double-quotes from the right-hand side (value). If you need to put a separator character (comma or vert-bar), you will need to escape them by inserting a backslash before them. Avoid its use if possible.

Syntax

```
MSet(name1=value1name2=value2)
```

Arguments

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

See Also

- [Asterisk 12 Application_Set](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_NBScat

NBScat()

Synopsis

Play an NBS local stream.

Description

Executes nbscat to listen to the local NBS stream. User can exit by pressing any key.

Syntax

```
NBScat ( )
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_NoCDR

NoCDR()

Synopsis

Tell Asterisk to not maintain a CDR for this channel.

Description

This application will tell Asterisk not to maintain a CDR for the current channel. This does **NOT** mean that information is not tracked; rather, if the channel is hung up no CDRs will be created for that channel.

If a subsequent call to ResetCDR occurs, all non-finalized CDRs created for the channel will be enabled.



Note

This application is deprecated. Please use the CDR_PROP function to disable CDRs on a channel.

Syntax

```
NoCDR( )
```

Arguments

See Also

- [Asterisk 12 Application_ResetCDR](#)
- [Asterisk 12 Function_CDR_PROP](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_NoOp

NoOp()

Synopsis

Do Nothing (No Operation).

Description

This application does nothing. However, it is useful for debugging purposes.

This method can be used to see the evaluations of variables or functions without having any effect.

Syntax

```
NoOp(text)
```

Arguments

- `text` - Any text provided can be viewed at the Asterisk CLI.

See Also

- [Asterisk 12 Application_Verbose](#)
- [Asterisk 12 Application_Log](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_Originate

Originate()

Synopsis

Originate a call.

Description

This application originates an outbound call and connects it to a specified extension or application. This application will block until the outgoing call fails or gets answered. At that point, this application will exit with the status variable set and dialplan processing will continue.

This application sets the following channel variable before exiting:

- `ORIGINATE_STATUS` - This indicates the result of the call origination.
 - `FAILED`
 - `SUCCESS`
 - `BUSY`
 - `CONGESTION`
 - `HANGUP`
 - `RINGING`
 - `UNKNOWN` - In practice, you should never see this value. Please report it to the issue tracker if you ever see it.

Syntax

```
Originate(tech_data,type,arg1[,arg2[,arg3[,timeout]]])
```

Arguments

- `tech_data` - Channel technology and data for creating the outbound channel. For example, SIP/1234.
- `type` - This should be `app` or `exten`, depending on whether the outbound channel should be connected to an application or extension.
- `arg1` - If the type is `app`, then this is the application name. If the type is `exten`, then this is the context that the channel will be sent to.
- `arg2` - If the type is `app`, then this is the data passed as arguments to the application. If the type is `exten`, then this is the extension that the channel will be sent to.
- `arg3` - If the type is `exten`, then this is the priority that the channel is sent to. If the type is `app`, then this parameter is ignored.
- `timeout` - Timeout in seconds. Default is 30 seconds.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_OSPAuth

OSPAuth()

Synopsis

OSP Authentication.

Description

Authenticate a call by OSP.

Input variables:

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

Output variables:

- `OSPINHANDLE` - The inbound call OSP transaction handle.
 - `OSPINLIMIT` - The inbound call duration limit in seconds.
- This application sets the following channel variable upon completion:

- `OSPAUTHSTATUS` - The status of OSPAuth attempt as a text string, one of
 - `SUCCESS`
 - `FAILED`
 - `ERROR`

Syntax

```
OSPAuth(provider,options)
```

Arguments

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

See Also

- [Asterisk 12 Application_OSPLookup](#)
- [Asterisk 12 Application_OSPNext](#)
- [Asterisk 12 Application_OSPFinish](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_OSPFinish

OSPFinish()

Synopsis

Report OSP entry.

Description

Report call state.

Input variables:

- `OSPINHANDLE` - The inbound call OSP transaction handle.
- `OSPOUTHANDLE` - The outbound call OSP transaction handle.
- `OSPAUTHSTATUS` - The OSPAuth status.
- `OSPLOOKUPSTATUS` - The OSPLookup status.
- `OSPNEXTSTATUS` - The OSPNext status.
- `OSPINAUDIOQOS` - The inbound call leg audio QoS string.
- `OSPOUTAUDIOQOS` - The outbound call leg audio QoS string.

This application sets the following channel variable upon completion:

- `OSPFINISHSTATUS` - The status of the OSPFinish attempt as a text string, one of
 - `SUCCESS`
 - `FAILED`
 - `ERROR`

Syntax

```
OSPFinish(cause,options)
```

Arguments

- `cause` - Hangup cause.
- `options` - Reserved.

See Also

- [Asterisk 12 Application_OSPAuth](#)
- [Asterisk 12 Application_OSPLookup](#)
- [Asterisk 12 Application_OSPNext](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_OSPLookup

OSPLookup()

Synopsis

Lookup destination by OSP.

Description

Looks up destination via OSP.

Input variables:

- **OSPINACTUALSRC** - The actual source device IP address in indirect mode.
- **OSPINPEERIP** - The last hop IP address.
- **OSPINTECH** - The inbound channel technology for the call.
- **OSPINHANDLE** - The inbound call OSP transaction handle.
- **OSPINTIMELIMIT** - The inbound call duration limit in seconds.
- **OSPINNETWORKID** - The inbound source network ID.
- **OSPINNPRN** - The inbound routing number.
- **OSPINNPCIC** - The inbound carrier identification code.
- **OSPINNPDI** - The inbound number portability database dip indicator.
- **OSPINSPID** - The inbound service provider identity.
- **OSPINOCN** - The inbound operator company number.
- **OSPINSPN** - The inbound service provider name.
- **OSPINALTSPN** - The inbound alternate service provider name.
- **OSPINMCC** - The inbound mobile country code.
- **OSPINMNC** - The inbound mobile network code.
- **OSPINTOHOST** - The inbound To header host part.
- **OSPINRPIDUSER** - The inbound Remote-Party-ID header user part.
- **OSPINPAUSER** - The inbound P-Asserted-Identify header user part.
- **OSPINDIVUSER** - The inbound Diversion header user part.
- **OSPINDIVHOST** - The inbound Diversion header host part.
- **OSPINPCIUSER** - The inbound P-Charge-Info header user part.
- **OSPINCUSTOMINFON** - The inbound custom information, where n is the index beginning with 1 upto 8.

Output variables:

- **OSPOUTHANDLE** - The outbound call OSP transaction handle.
- **OSPOUTTECH** - The outbound channel technology for the call.
- **OSPDESTINATION** - The outbound destination IP address.
- **OSPOUTCALLING** - The outbound calling number.
- **OSPOUTCALLED** - The outbound called number.
- **OSPOUTNETWORKID** - The outbound destination network ID.
- **OSPOUTNPRN** - The outbound routing number.
- **OSPOUTNPCIC** - The outbound carrier identification code.
- **OSPOUTNPDI** - The outbound number portability database dip indicator.
- **OSPOUTSPID** - The outbound service provider identity.
- **OSPOUTOCN** - The outbound operator company number.
- **OSPOUTSPN** - The outbound service provider name.
- **OSPOUTALTSPN** - The outbound alternate service provider name.
- **OSPOUTMCC** - The outbound mobile country code.
- **OSPOUTMNC** - The outbound mobile network code.
- **OSPOUTTOKEN** - The outbound OSP token.
- **OSPDESTREMAINS** - The number of remained destinations.
- **OSPOUTTIMELIMIT** - The outbound call duration limit in seconds.
- **OSPOUTCALLIDTYPES** - The outbound Call-ID types.
- **OSPOUTCALLID** - The outbound Call-ID. Only for H.323.
- **OSPDIALSTR** - The outbound Dial command string.

This application sets the following channel variable upon completion:

- `OSPLookupStatus` - The status of OSPLookup attempt as a text string, one of
 - `SUCCESS`
 - `FAILED`
 - `ERROR`

Syntax

```
OSPLookup(exten,provider,options)
```

Arguments

- `exten` - The exten of the call.
- `provider` - The name of the provider that is used to route the call.
- `options`
 - `h` - generate H323 call id for the outbound call
 - `s` - generate SIP call id for the outbound call. Have not been implemented
 - `i` - generate IAX call id for the outbound call. Have not been implemented

See Also

- [Asterisk 12 Application_OSPAuth](#)
- [Asterisk 12 Application_OSPNext](#)
- [Asterisk 12 Application_OSPFinish](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_OSPNext

OSPNext()

Synopsis

Lookup next destination by OSP.

Description

Looks up the next destination via OSP.

Input variables:

- `OSPINHANDLE` - The inbound call OSP transaction handle.
- `OSPOUTHANDLE` - The outbound call OSP transaction handle.
- `OSPINTIMELIMIT` - The inbound call duration limit in seconds.
- `OSPOUTCALLIDTYPES` - The outbound Call-ID types.
- `OSPDESTREMAILS` - The number of remained destinations.

Output variables:

- `OSPOUTTECH` - The outbound channel technology.
- `OSPDESTINATION` - The destination IP address.
- `OSPOUTCALLING` - The outbound calling number.
- `OSPOUTCALLED` - The outbound called number.
- `OSPOUTNETWORKID` - The outbound destination network ID.
- `OSPOUTNPRN` - The outbound routing number.
- `OSPOUTNPCIC` - The outbound carrier identification code.
- `OSPOUTNPDI` - The outbound number portability database dip indicator.
- `OSPOUTSPID` - The outbound service provider identity.
- `OSPOUTOCN` - The outbound operator company number.
- `OSPOUTSPN` - The outbound service provider name.
- `OSPOUTALTSPN` - The outbound alternate service provider name.
- `OSPOUTMCC` - The outbound mobile country code.
- `OSPOUTMNC` - The outbound mobile network code.
- `OSPOUTTOKEN` - The outbound OSP token.
- `OSPDESTREMAILS` - The number of remained destinations.
- `OSPOUTTIMELIMIT` - The outbound call duration limit in seconds.
- `OSPOUTCALLID` - The outbound Call-ID. Only for H.323.
- `OSPDIALSTR` - The outbound Dial command string.

This application sets the following channel variable upon completion:

- `OSPNEXTSTATUS` - The status of the OSPNext attempt as a text string, one of
 - `SUCCESS`
 - `FAILED`
 - `ERROR`

Syntax

See Also

- [Asterisk 12 Application_OSPAuth](#)
- [Asterisk 12 Application_OSPLookup](#)
- [Asterisk 12 Application_OSPFinish](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Page

Page()

Synopsis

Page series of phones


Description

Places outbound calls to the given *technology/resource* and dumps them into a conference bridge as muted participants. The original caller is dumped into the conference as a speaker and the room is destroyed when the original caller leaves.

Syntax

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

Arguments

- **Technology/Resource**
 - **Technology/Resource** - Specification of the device(s) to dial. These must be in the format of *Technology/Resource*, where *Technology* represents a particular channel driver, and *Resource* represents a resource available to that particular channel driver.
 - **Technology2/Resource2** - Optional extra devices to dial in parallel
If you need more than one, enter them as *Technology2/Resource2& Technology3/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 (*CONFBRIDGE(bridge,record_conference)*)
 - **s** - Only dial a channel if its device state says that it is *NOT_INUSE*
 - **A** - Play an announcement to all paged participants
 - **x** - The announcement to playback to all devices
 - **n** - Do not play announcement to caller (alters **A**  behavior)
- **timeout** - Specify the length of time that the system will attempt to connect a call. After this duration, any page calls that have not been answered will be hung up by the system.

See Also

- [Asterisk 12 Application_ConfBridge](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Park

Park()

Synopsis

Park yourself.

Description

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

If you set the `PARKINGEXTEN` variable to a parking space extension in the parking lot, `Park()` will attempt to park the call on that extension. If the extension is already in use then execution will continue at the next priority.

Syntax

```
Park(parking_lot_name,options)
```

Arguments

- `parking_lot_name` - Specify in which parking lot to park a call.
The parking lot used is selected in the following order:
 - 1) `parking_lot_name` option to this application
 - 2) `PARKINGLOT` variable
 - 3) `CHANNEL(parkinglot)` function (Possibly preset by the channel driver.)
 - 4) Default parking lot.
- `options` - A list of options for this parked call.
 - `r` - Send ringing instead of MOH to the parked call.
 - `R` - Randomize the selection of a parking space.
 - `s` - Silence announcement of the parking space number.
 - `c` - If the parking times out, go to this place in the dialplan instead of where the parking lot defines the call should go.
 - `context`
 - `extension`
 - `priority`
 - `t` - Use a timeout of `duration` seconds instead of the timeout specified by the parking lot.
 - `duration`

See Also

- [Asterisk 12 Application_ParkedCall](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ParkAndAnnounce

ParkAndAnnounce()

Synopsis

Park and Announce.

Description

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

The variable `PARKEDAT` will contain the parking extension into which the call was placed. Use with the Local channel to allow the dialplan to make use of this information.

Syntax

```
ParkAndAnnounce(parking_lot_name,optionsannounce:announce1[:...],dial)
```

Arguments

- `parking_lot_name` - Specify in which parking lot to park a call.
The parking lot used is selected in the following order:
 - 1) `parking_lot_name` option to this application
 - 2) `PARKINGLOT` variable
 - 3) `CHANNEL(parkinglot)` function (Possibly preset by the channel driver.)
 - 4) Default parking lot.
- `options` - A list of options for this parked call.
 - `r` - Send ringing instead of MOH to the parked call.
 - `R` - Randomize the selection of a parking space.
 - `c` - If the parking times out, go to this place in the dialplan instead of where the parking lot defines the call should go.
 - `context`
 - `extension`
 - `priority`
 - `t` - Use a timeout of `duration` seconds instead of the timeout specified by the parking lot.
 - `duration`
- `announce_template`
 - `announce` - Colon-separated list of files to announce. The word `PARKED` will be replaced by a `say_digits` of the extension in which the call is parked.
 - `announce1`
- `dial` - The `app_dial` style resource to call to make the announcement. Console/dsp calls the console.

See Also

- [Asterisk 12 Application_Park](#)
- [Asterisk 12 Application_ParkedCall](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ParkedCall

ParkedCall()

Synopsis

Retrieve a parked call.

Description

Used to retrieve a parked call from a parking lot.



Note

If a parking lot's parkext option is set, then Parking lots will automatically create and manage dialplan extensions in the parking lot context. If that is the case then you will not need to manage parking extensions yourself, just include the parking context of the parking lot.

Syntax

```
ParkedCall(parking_lot_name,parking_space)
```

Arguments

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

See Also

- [Asterisk 12 Application_Park](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_UnpauseMonitor](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_PauseQueueMember

PauseQueueMember()

Synopsis

Pauses a queue member.

Description

Pauses (blocks calls for) a queue member. The given interface will be paused in the given queue. This prevents any calls from being sent from the queue to the interface until it is unpaused with `UnpauseQueueMember` or the manager interface. If no queue name 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(queue_name, 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 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Pickup

Pickup()

Synopsis

Directed extension call pickup.

Description

This application can pickup a specified ringing channel. The channel to pickup can be specified in the following ways.

- 1) If no *extension* targets are specified, the application will pickup a channel matching the pickup group of the requesting channel.
- 2) If the *extension* is specified with a *context* of the special string `PICKUPMARK` (for example `10@PICKUPMARK`), the application will pickup a channel which has defined the channel variable `PICKUPMARK` with the same value as *extension* (in this example, `10`).
- 3) If the *extension* is specified with or without a *context*, the channel with a matching *extension* and *context* will be picked up. If no *context* is specified, the current context will be used.



Note

The *extension* is typically set on matching channels by the dial application that created the channel. The *context* is set on matching channels by the channel driver for the device.

Syntax

```
Pickup(extension&extension2[&...])
```

Arguments

- targets
 - *extension* - Specification of the pickup target.
 - *extension*
 - *context*
 - *extension2* - Additional specifications of pickup targets.
 - *extension2*
 - *context2*

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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-existent it will fail

This application sets the following channel variable upon completion:

- `PLAYBACKSTATUS` - The status of the playback attempt as a text string.
 - `SUCCESS`
 - `FAILED`
- See Also: `Background` (application) – for playing sound files that are interruptible

`WaitExten` (application) – wait for digits from caller, optionally play music on hold

Syntax

```
Playback(filename&filename2[&...],options)
```

Arguments

- `filenames`
 - `filename`
 - `filename2`
- `options` - Comma separated list of options
 - `skip` - Do not play if not answered
 - `noanswer` - Playback without answering, otherwise the channel will be answered before the sound is played.

See Also

- [Asterisk 12 Application_Background](#)
- [Asterisk 12 Application_WaitExten](#)
- [Asterisk 12 Application_ControlPlayback](#)
- [Asterisk 12 AGICommand_stream file](#)
- [Asterisk 12 AGICommand_control stream file](#)
- [Asterisk 12 ManagerAction_ControlPlayback](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_StopPlayTones](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_Zapateller](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Proceeding

Proceeding()

Synopsis

Indicate proceeding.

Description

This application will request that a proceeding message be provided to the calling channel.

Syntax

```
Proceeding()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_Busy](#)
- [Asterisk 12 Application_Congestion](#)
- [Asterisk 12 Application_Ringing](#)
- [Asterisk 12 Application_Playtones](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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(queueename,options,URL,announceoverride,timeout,AGI,macro,gosub,rule,position)
```

Arguments

- **queueename**
- **options**
 - **C** - Mark all calls as "answered elsewhere" when cancelled.
 - **c** - Continue in the dialplan if the callee hangs up.
 - **d** - data-quality (modem) call (minimum delay).
 - **F** - When the caller hangs up, transfer the **called member** to the specified destination and **start** execution at that location.
 - context
 - exten
 - priority
 - **F** - When the caller hangs up, transfer the **called member** to the next priority of the current extension and **start** execution at that location.
 - **h** - Allow **callee** to hang up by pressing *.
 - **H** - Allow **caller** to hang up by pressing *.
 - **n** - No retries on the timeout; will exit this application and go to the next step.
 - **i** - Ignore call forward requests from queue members and do nothing when they are requested.
 - **I** - Asterisk will ignore any connected line update requests or any redirecting party update requests it may receive on this dial attempt.
 - **r** - Ring instead of playing MOH. Periodic Announcements are still made, if applicable.
 - **R** - Ring instead of playing MOH when a member channel is actually ringing.
 - **t** - Allow the **called** user to transfer the calling user.
 - **T** - Allow the **calling** user to transfer the call.
 - **w** - Allow the **called** user to write the conversation to disk via Monitor.
 - **W** - Allow the **calling** user to write the conversation to disk via Monitor.
 - **k** - Allow the **called** party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.conf`.
 - **K** - Allow the **calling** party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.conf`.
 - **x** - Allow the **called** user to write the conversation to disk via MixMonitor.

- `x` - Allow the **calling** user to write the conversation to disk via MixMonitor.
- `URL` - URL will be sent to the called party if the channel supports it.
- `announceoverride`
- `timeout` - Will cause the queue to fail out after a specified number of seconds, checked between each `queues.conf` *timeout* and *retry* cycle.
- `AGI` - Will setup an AGI script to be executed on the calling party's channel once they are connected to a queue member.
- `macro` - Will run a macro on the calling party's channel once they are connected to a queue member.
- `gosub` - Will run a gosub on the calling party's channel once they are connected to a queue member.
- `rule` - Will cause the queue's default rule 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 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Function_Exception](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Read

Read()

Synopsis

Read a variable.

Description

Reads a #-terminated string of digits a certain number of times from the user in to the given *variable*.

This application sets the following channel variable upon completion:

- **READSTATUS** - This is the status of the read operation.
 - OK
 - ERROR
 - HANGUP
 - INTERRUPTED
 - SKIPPED
 - TIMEOUT

Syntax

```
Read(variablefilename&filename2[&...],maxdigits,options,attempts,timeout)
```

Arguments

- **variable** - The input digits will be stored in the given *variable* name.
- **filenames**
 - **filename** - file(s) to play before reading digits or tone with option **i**
 - **filename2**
- **maxdigits** - Maximum acceptable number of digits. Stops reading after *maxdigits* have been entered (without requiring the user to press the # key).
Defaults to 0 - no limit - wait for the user press the # key. Any value below 0 means the same. Max accepted value is 255.
- **options**
 - **s** - to return immediately if the line is not up.
 - **i** - to play filename as an indication tone from your `indications.conf`.
 - **n** - to read digits even if the line is not up.
- **attempts** - If greater than 1, that many *attempts* will be made in the event no data is entered.
- **timeout** - The number of seconds to wait for a digit response. If greater than 0, that value will override the default timeout. Can be floating point.

See Also

- [Asterisk 12 Application_SendDTMF](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ReadExten

ReadExten()

Synopsis

Read an extension into a variable.

Description

Reads a # terminated string of digits from the user into the given variable.

Will set READEXTENSTATUS on exit with one of the following statuses:

- READEXTENSTATUS
 - OK - A valid extension exists in \${variable}.
 - TIMEOUT - No extension was entered in the specified time. Also sets \${variable} to "t".
 - INVALID - An invalid extension, \${INVALID_EXTEN}, was entered. Also sets \${variable} to "i".
 - SKIP - Line was not up and the option 's' was specified.
 - ERROR - Invalid arguments were passed.

Syntax

```
ReadExten(variable,filename,context,option,timeout)
```

Arguments

- `variable`
- `filename` - File to play before reading digits or tone with option `i`
- `context` - Context in which to match extensions.
- `option`
 - `s` - Return immediately if the channel is not answered.
 - `i` - Play *filename* as an indication tone from your `indications.conf` or a directly specified list of frequencies and durations.
 - `n` - Read digits even if the channel is not answered.
- `timeout` - An integer number of seconds to wait for a digit response. If greater than 0, that value will override the default timeout.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ReadFile

ReadFile()

Synopsis

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

Description

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



Warning

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

Syntax

```
ReadFile(varnamefile[length])
```

Arguments

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

See Also

- [Asterisk 12 Application_System](#)
- [Asterisk 12 Application_Read](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ReceiveFAX_app_fax

ReceiveFAX() - [app_fax]

Synopsis

Receive a Fax

Description

Receives a FAX from the channel into the given filename overwriting the file if it already exists.

File created will be in TIFF format.

This application sets the following channel variables:

- LOCALSTATIONID - To identify itself to the remote end
- LOCALHEADERINFO - To generate a header line on each page
- FAXSTATUS
 - SUCCESS
 - FAILED
- FAXERROR - Cause of failure
- REMOTESTATIONID - The CSID of the remote side
- FAXPAGES - Number of pages sent
- FAXBITRATE - Transmission rate
- FAXRESOLUTION - Resolution of sent fax

Syntax

```
ReceiveFAX(filename[,c])
```

Arguments

- filename - Filename of TIFF file save incoming fax
- c - Makes the application behave as the calling machine
(Default behavior is as answering machine)

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ReceiveFAX_res_fax

ReceiveFAX() - [res_fax]

Synopsis

Receive a FAX and save as a TIFF/F file.

Description

This application is provided by res_fax, which is a FAX technology agnostic module that utilizes FAX technology resource modules to complete a FAX transmission.

Session arguments can be set by the FAXOPT function and to check results of the ReceiveFax() application.

Syntax

```
ReceiveFAX(filename,options)
```

Arguments

- `filename`
- `options`
 - `d` - Enable FAX debugging.
 - `f` - Allow audio fallback FAX transfer on T.38 capable channels.
 - `F` - Force usage of audio mode on T.38 capable channels.
 - `s` - Send progress Manager events (overrides statusevents setting in res_fax.conf).

See Also

- [Asterisk 12 Function_FAXOPT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Record

Record()

Synopsis

Record to a file.

Description

If filename contains %d, these characters will be replaced with a number incremented by one each time the file is recorded. Use `core show file formats` to see the available formats on your system. User can press # to terminate the recording and continue to the next priority. If the user hangs up during a recording, all data will be lost and the application will terminate.

- `RECORDED_FILE` - Will be set to the final filename of the recording.
- `RECORD_STATUS` - This is the final status of the command
 - `DTMF` - A terminating DTMF was received ('#' or '*', depending upon option 't')
 - `SILENCE` - The maximum silence occurred in the recording.
 - `SKIP` - The line was not yet answered and the 's' option was specified.
 - `TIMEOUT` - The maximum length was reached.
 - `HANGUP` - The channel was hung up.
 - `ERROR` - An unrecoverable error occurred, which resulted in a `WARNING` to the logs.

Syntax

```
Record(filename.format,silence,maxduration,options)
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_RemoveQueueMember

RemoveQueueMember()

Synopsis

Dynamically removes queue members.

Description

If the interface is **NOT** in the queue it will return an error.

This application sets the following channel variable upon completion:

- RQMSTATUS
 - REMOVED
 - NOTINQUEUE
 - NOSUCHQUEUE
 - NOTDYNAMIC
- Example: RemoveQueueMember(techsupport,SIP/3000)

Syntax

```
RemoveQueueMember ( queueName , interface )
```

Arguments

- queueName
- interface

See Also

- [Asterisk 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_ResetCDR

ResetCDR()

Synopsis

Resets the Call Data Record.

Description

This application causes the Call Data Record to be reset. Depending on the flags passed in, this can have several effects. With no options, a reset does the following:

1. The `start` time is set to the current time.
2. If the channel is answered, the `answer` time is set to the current time.
3. All variables are wiped from the CDR. Note that this step can be prevented with the `v` option.

On the other hand, if the `e` option is specified, the effects of the NoCDR application will be lifted. CDRs will be re-enabled for this channel.



Note

The `e` option is deprecated. Please use the `CDR_PROP` function instead.

Syntax

```
ResetCDR(options)
```

Arguments

- `options`
 - `v` - Save the CDR variables during the reset.
 - `e` - Enable the CDRs for this channel only (negate effects of NoCDR).

See Also

- [Asterisk 12 Application_ForkCDR](#)
- [Asterisk 12 Application_NoCDR](#)
- [Asterisk 12 Function_CDR_PROP](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_RetryDial

RetryDial()

Synopsis

Place a call, retrying on failure allowing an optional exit extension.

Description

This application will attempt to place a call using the normal Dial application. If no channel can be reached, the *announce* file will be played. Then, it will wait *sleep* number of seconds before retrying the call. After *retries* number of attempts, the calling channel will continue at the next priority in the dialplan. If the *retries* setting is set to 0, this application will retry endlessly. While waiting to retry a call, a 1 digit extension may be dialed. If that extension exists in either the context defined in *EXITCONTEXT* or the current one, The call will jump to that extension immediately. The *dialargs* are specified in the same format that arguments are provided to the Dial application.

Syntax

```
RetryDial(announce,sleep,retries,dialargs)
```

Arguments

- *announce* - Filename of sound that will be played when no channel can be reached
- *sleep* - Number of seconds to wait after a dial attempt failed before a new attempt is made
- *retries* - Number of retries
When this is reached flow will continue at the next priority in the dialplan
- *dialargs* - Same format as arguments provided to the Dial application

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Return

Return()

Synopsis

Return from gosub routine.

Description

Jumps to the last label on the stack, removing it. The return *value*, if any, is saved in the channel variable GOSUB_RETVAL.

Syntax

```
Return(value)
```

Arguments

- `value` - Return value.

See Also

- [Asterisk 12 Application_Gosub](#)
- [Asterisk 12 Application_StackPop](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Ringing

Ringing()

Synopsis

Indicate ringing tone.

Description

This application will request that the channel indicate a ringing tone to the user.

Syntax

```
Ringinɡ( )
```

Arguments

See Also

- [Asterisk 12 Application_Busy](#)
- [Asterisk 12 Application_Congestion](#)
- [Asterisk 12 Application_Progress](#)
- [Asterisk 12 Application_Playtones](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_SayDigits](#)
- [Asterisk 12 Application_SayNumber](#)
- [Asterisk 12 Application_SayPhonetic](#)
- [Asterisk 12 Function_CHANNEL](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SayAlphaCase

SayAlphaCase()

Synopsis

Say Alpha.

Description

This application will play the sounds that correspond to the letters of the given *string*. Optionally, a *casetype* may be specified. This will be used for case-insensitive or case-sensitive pronunciations.

Syntax

```
SayAlphaCase(casetype,string)
```

Arguments

- `casetype`
 - `a` - Case sensitive (all) pronunciation. (Ex: SayAlphaCase(a,aBc); - lowercase a uppercase b lowercase c).
 - `l` - Case sensitive (lower) pronunciation. (Ex: SayAlphaCase(l,aBc); - lowercase a b lowercase c).
 - `n` - Case insensitive pronunciation. Equivalent to SayAlpha. (Ex: SayAlphaCase(n,aBc) - a b c).
 - `u` - Case sensitive (upper) pronunciation. (Ex: SayAlphaCase(u,aBc); - a uppercase b c).
- `string`

See Also

- [Asterisk 12 Application_SayDigits](#)
- [Asterisk 12 Application_SayNumber](#)
- [Asterisk 12 Application_SayPhonetic](#)
- [Asterisk 12 Application_SayAlpha](#)
- [Asterisk 12 Function_CHANNEL](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 be 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 12 Application_SayCountedNoun](#)
- [Asterisk 12 Application_SayNumber](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 preceded 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 12 Application_SayCountedAdj](#)
- [Asterisk 12 Application_SayNumber](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 12 Application_SayAlpha](#)
- [Asterisk 12 Application_SayNumber](#)
- [Asterisk 12 Application_SayPhonetic](#)
- [Asterisk 12 Function_CHANNEL](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SayNumber

SayNumber()

Synopsis

Say Number.

Description

This application will play the sounds that correspond to the given *digits*. Optionally, a *gender* may be specified. This will use the language that is currently set for the channel. See the CHANNEL() function for more information on setting the language for the channel.

Syntax

```
SayNumber(digits,gender)
```

Arguments

- `digits`
- `gender`

See Also

- [Asterisk 12 Application_SayAlpha](#)
- [Asterisk 12 Application_SayDigits](#)
- [Asterisk 12 Application_SayPhonetic](#)
- [Asterisk 12 Function_CHANNEL](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SayPhonetic

SayPhonetic()

Synopsis

Say Phonetic.

Description

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

Syntax

```
SayPhonetic(string)
```

Arguments

- `string`

See Also

- [Asterisk 12 Application_SayAlpha](#)
- [Asterisk 12 Application_SayDigits](#)
- [Asterisk 12 Application_SayNumber](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SayUnixTime

SayUnixTime()

Synopsis

Says a specified time in a custom format.

Description

Uses some of the sound files stored in `/var/lib/asterisk/sounds` to construct a phrase saying the specified date and/or time in the specified format.

Syntax

```
SayUnixTime([unixtime[,timezone[,format[,options]]]])
```

Arguments

- `unixtime` - time, in seconds since Jan 1, 1970. May be negative. Defaults to now.
- `timezone` - timezone, see `/usr/share/zoneinfo` for a list. Defaults to machine default.
- `format` - a format the time is to be said in. See `voicemail.conf`. Defaults to `ABdY "digits/at" IMp`
- `options`
 - `j` - Allow the calling user to dial digits to jump to that extension.

See Also

- [Asterisk 12 Function_STRFTIME](#)
- [Asterisk 12 Function_STRPTIME](#)
- [Asterisk 12 Function_IFTIME](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SendDTMF

SendDTMF()

Synopsis

Sends arbitrary DTMF digits

Description

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

Syntax

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

Arguments

- `digits` - List of digits 0-9,*,#,a-d,A-D to send also w for a half second pause, W for a one second pause, and f or F for a flash-hook if the channel supports flash-hook.
- `timeout_ms` - Amount of time to wait in ms between tones. (defaults to .25s)
- `duration_ms` - Duration of each digit
- `channel` - Channel where digits will be played

See Also

- [Asterisk 12 Application_Read](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SendFAX_app_fax

SendFAX() - [app_fax]

Synopsis

Send a Fax

Description

Send a given TIFF file to the channel as a FAX.

This application sets the following channel variables:

- LOCALSTATIONID - To identify itself to the remote end
- LOCALHEADERINFO - To generate a header line on each page
- FAXSTATUS
 - SUCCESS
 - FAILED
- FAXERROR - Cause of failure
- REMOTESTATIONID - The CSID of the remote side
- FAXPAGES - Number of pages sent
- FAXBITRATE - Transmission rate
- FAXRESOLUTION - Resolution of sent fax

Syntax

```
SendFAX(filename[,a])
```

Arguments

- filename - Filename of TIFF file to fax
- a - Makes the application behave as the answering machine
(Default behavior is as calling machine)

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SendFAX_res_fax

SendFAX() - [res_fax]

Synopsis

Sends a specified TIFF/F file as a FAX.

Description

This application is provided by res_fax, which is a FAX technology agnostic module that utilizes FAX technology resource modules to complete a FAX transmission.

Session arguments can be set by the FAXOPT function and to check results of the SendFax() application.

Syntax

```
SendFAX(filename2[&...],options)
```

Arguments

- `filename`
 - `filename2` - TIFF file to send as a FAX.
- `options`
 - `d` - Enable FAX debugging.
 - `f` - Allow audio fallback FAX transfer on T.38 capable channels.
 - `F` - Force usage of audio mode on T.38 capable channels.
 - `s` - Send progress Manager events (overrides `statusevents` setting in `res_fax.conf`).
 - `z` - Initiate a T.38 reinvite on the channel if the remote end does not.

See Also

- [Asterisk 12 Function_FAXOPT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SendImage

SendImage()

Synopsis

Sends an image file.

Description

Send an image file on a channel supporting it.

Result of transmission will be stored in `SENDIMAGESTATUS`

- `SENDIMAGESTATUS`
 - `SUCCESS` - Transmission succeeded.
 - `FAILURE` - Transmission failed.
 - `UNSUPPORTED` - Image transmission not supported by channel.

Syntax

```
SendImage(filename)
```

Arguments

- `filename` - Path of the filename (image) to send.

See Also

- [Asterisk 12 Application_SendText](#)
- [Asterisk 12 Application_SendURL](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SendText

SendText()

Synopsis

Send a Text Message.

Description

Sends *text* to current channel (callee).

Result of transmission will be stored in the `SENDEXTSTATUS`

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



Note

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

Syntax

```
SendText ( text )
```

Arguments

- `text`

See Also

- [Asterisk 12 Application_SendImage](#)
- [Asterisk 12 Application_SendURL](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SendURL

SendURL()

Synopsis

Send a URL.

Description

Requests client go to *URL* (IAX2) or sends the URL to the client (other channels).

Result is returned in the `SENDURLSTATUS` channel variable:

- `SENDURLSTATUS`
 - `SUCCESS` - URL successfully sent to client.
 - `FAILURE` - Failed to send URL.
 - `NOLOAD` - Client failed to load URL (wait enabled).
 - `UNSUPPORTED` - Channel does not support URL transport.
SendURL continues normally if the URL was sent correctly or if the channel does not support HTML transport. Otherwise, the channel is hung up.

Syntax

```
SendURL(URL,option)
```

Arguments

- `URL`
- `option`
 - `w` - Execution will wait for an acknowledgement that the URL has been loaded before continuing.

See Also

- [Asterisk 12 Application_SendImage](#)
- [Asterisk 12 Application_SendText](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Set

Set()

Synopsis

Set channel variable or function value.

Description

This function can be used to set the value of channel variables or dialplan functions. When setting variables, if the variable name is prefixed with `_`, the variable will be inherited into channels created from the current channel. If the variable name is prefixed with `__`, the variable will be inherited into channels created from the current channel and all children channels.



Note

If (and only if), in `/etc/asterisk/asterisk.conf`, you have a `[compat]` category, and you have `app_set = 1.4` under that, then the behavior of this app changes, and strips surrounding quotes from the right hand side as it did previously in 1.4. The advantages of not stripping out quoting, and not caring about the separator characters (comma and vertical bar) were sufficient to make these changes in 1.6. Confusion about how many backslashes would be needed to properly protect separators and quotes in various database access strings has been greatly reduced by these changes.

Syntax

```
Set(name=value)
```

Arguments

- `name`
- `value`

See Also

- [Asterisk 12 Application_MSet](#)
- [Asterisk 12 Function_GLOBAL](#)
- [Asterisk 12 Function_SET](#)
- [Asterisk 12 Function_ENV](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SetAMAFlags

SetAMAFlags()

Synopsis

Set the AMA Flags.

Description

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



Warning

This application is deprecated. Please use the CHANNEL function instead.

Syntax

```
SetAMAFlags(flag)
```

Arguments

- flag

See Also

- [Asterisk 12 Function_CDR](#)
- [Asterisk 12 Function_CHANNEL](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_SIPSendCustomINFO

SIPSendCustomINFO()

Synopsis

Send a custom INFO frame on specified channels.

Description

SIPSendCustomINFO() allows you to send a custom INFO message on all active SIP channels or on channels with the specified User Agent. This application is only available if TEST_FRAMEWORK is defined.

Syntax

```
SIPSendCustomINFO(Data[ ,UserAgent ])
```

Arguments

- Data
- UserAgent

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SkelGuessNumber

SkelGuessNumber()

Synopsis

An example number guessing game

Description

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

Syntax

```
SkelGuessNumber(level,options)
```

Arguments

- `level`
- `options`
 - `c` - The computer should cheat
 - `n` - How many games to play before hanging up

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SLAStation

SLAStation()

Synopsis

Shared Line Appearance Station.

Description

This application should be executed by an SLA station. The argument depends on how the call was initiated. If the phone was just taken off hook, then the argument *station* should be just the station name. If the call was initiated by pressing a line key, then the station name should be preceded by an underscore and the trunk name associated with that line button.

For example: `station1_line1`

On exit, this application will set the variable `SLASTATION_STATUS` to one of the following values:

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

Syntax

```
SLAStation(station)
```

Arguments

- `station` - Station name

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SLATrunk

SLATrunk()

Synopsis

Shared Line Appearance Trunk.

Description

This application should be executed by an SLA trunk on an inbound call. The channel calling this application should correspond to the SLA trunk with the name *trunk* that is being passed as an argument.

On exit, this application will set the variable `SLATRUNK_STATUS` to one of the following values:

- `SLATRUNK_STATUS`
 - `FAILURE`
 - `SUCCESS`
 - `UNANSWERED`
 - `RINGTIMEOUT`

Syntax

```
SLATrunk(trunk,options)
```

Arguments

- `trunk` - Trunk name
- `options`
 - `M` - Play back the specified MOH *class* instead of ringing
 - `class`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SMS

SMS()

Synopsis

Communicates with SMS service centres and SMS capable analogue phones.

Description

SMS handles exchange of SMS data with a call to/from SMS capable phone or SMS PSTN service center. Can send and/or receive SMS messages. Works to ETSI ES 201 912; compatible with BT SMS PSTN service in UK and Telecom Italia in Italy.

Typical usage is to use to handle calls from the SMS service centre CLI, or to set up a call using `outgoing` or `manager` interface to connect service centre to SMS().

"Messages are processed as per text file message queues. `smsq` (a separate software) is a command to generate message queues and send messages.



Note

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

Syntax

```
SMS(name,options,addr,body)
```

Arguments

- `name` - The name of the queue used in `/var/spool/asterisk/sms`
- `options`
 - `a` - Answer, i.e. send initial FSK packet.
 - `s` - Act as service centre talking to a phone.
 - `t` - Use protocol 2 (default used is protocol 1).
 - `p` - Set the initial delay to N ms (default is 300). `addr` and `body` are a deprecated format to send messages out.
 - `r` - Set the Status Report Request (SRR) bit.
 - `o` - The body should be coded as octets not 7-bit symbols.
- `addr`
- `body`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_SpeechActivateGrammar

SpeechActivateGrammar()

Synopsis

Activate a grammar.

Description

This activates the specified grammar to be recognized by the engine. A grammar tells the speech recognition engine what to recognize, and how to portray it back to you in the dialplan. The grammar name is the only argument to this application.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechActivateGrammar(grammar_name)
```

Arguments

- grammar_name

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SpeechBackground

SpeechBackground()

Synopsis

Play a sound file and wait for speech to be recognized.

Description

This application plays a sound file and waits for the person to speak. Once they start speaking playback of the file stops, and silence is heard. Once they stop talking the processing sound is played to indicate the speech recognition engine is working. Once results are available the application returns and results (score and text) are available using dialplan functions.

The first text and score are \${SPEECH_TEXT(0)} AND \${SPEECH_SCORE(0)} while the second are \${SPEECH_TEXT(1)} and \${SPEECH_SCORE(1)}.

The first argument is the sound file and the second is the timeout integer in seconds.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechBackground(sound_file,timeout,options)
```

Arguments

- `sound_file`
- `timeout` - Timeout integer in seconds. Note the timeout will only start once the sound file has stopped playing.
- `options`
 - `n` - Don't answer the channel if it has not already been answered.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SpeechCreate

SpeechCreate()

Synopsis

Create a Speech Structure.

Description

This application creates information to be used by all the other applications. It must be called before doing any speech recognition activities such as activating a grammar. It takes the engine name to use as the argument, if not specified the default engine will be used.

Sets the ERROR channel variable to 1 if the engine cannot be used.

Syntax

```
SpeechCreate(engine_name)
```

Arguments

- engine_name

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SpeechDeactivateGrammar

SpeechDeactivateGrammar()

Synopsis

Deactivate a grammar.

Description

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

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechDeactivateGrammar(grammar_name)
```

Arguments

- `grammar_name` - The grammar name to deactivate

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SpeechDestroy

SpeechDestroy()

Synopsis

End speech recognition.

Description

This destroys the information used by all the other speech recognition applications. If you call this application but end up wanting to recognize more speech, you must call SpeechCreate() again before calling any other application.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechDestroy()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SpeechLoadGrammar

SpeechLoadGrammar()

Synopsis

Load a grammar.

Description

Load a grammar only on the channel, not globally.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechLoadGrammar(grammar_name,path)
```

Arguments

- grammar_name
- path

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SpeechProcessingSound

SpeechProcessingSound()

Synopsis

Change background processing sound.

Description

This changes the processing sound that SpeechBackground plays back when the speech recognition engine is processing and working to get results.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechProcessingSound(sound_file)
```

Arguments

- sound_file

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SpeechStart

SpeechStart()

Synopsis

Start recognizing voice in the audio stream.

Description

Tell the speech recognition engine that it should start trying to get results from audio being fed to it.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechStart()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_SpeechUnloadGrammar

SpeechUnloadGrammar()

Synopsis

Unload a grammar.

Description

Unload a grammar.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechUnloadGrammar(grammar_name)
```

Arguments

- grammar_name

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_StackPop

StackPop()

Synopsis

Remove one address from gosub stack.

Description

Removes last label on the stack, discarding it.

Syntax

```
StackPop ( )
```

Arguments

See Also

- [Asterisk 12 Application_Return](#)
- [Asterisk 12 Application_Gosub](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_StartMusicOnHold

StartMusicOnHold()

Synopsis

Play Music On Hold.

Description

Starts playing music on hold, uses default music class for channel. Starts playing music specified by class. If omitted, the default music source for the channel will be used. Always returns 0.

Syntax

```
StartMusicOnHold(class)
```

Arguments

- class

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Stasis

Stasis()

Synopsis

Invoke an external Stasis application.

Description

Invoke a Stasis application.

Syntax

```
Stasis(app_name,args)
```

Arguments

- `app_name` - Name of the application to invoke.
- `args` - Optional comma-delimited arguments for the application invocation.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_StopMixMonitor

StopMixMonitor()

Synopsis

Stop recording a call through MixMonitor, and free the recording's file handle.

Description

Stops the audio recording that was started with a call to `MixMonitor()` on the current channel.

Syntax

```
StopMixMonitor([MixMonitorID])
```

Arguments

- `MixMonitorID` - If a valid ID is provided, then this command will stop only that specific MixMonitor.

See Also

- [Asterisk 12 Application_MixMonitor](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_StopMusicOnHold

StopMusicOnHold()

Synopsis

Stop playing Music On Hold.

Description

Stops playing music on hold.

Syntax

```
StopMusicOnHold()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_StopPlayTones

StopPlayTones()

Synopsis

Stop playing a tone list.

Description

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

Syntax

```
StopPlayTones ( )
```

Arguments

See Also

- [Asterisk 12 Application_PlayTones](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_System

System()

Synopsis

Execute a system command.

Description

Executes a command by using system(). If the command fails, the console should report a fallback.

Result of execution is returned in the SYSTEMSTATUS channel variable:

- SYSTEMSTATUS
 - FAILURE - Could not execute the specified command.
 - SUCCESS - Specified command successfully executed.

Syntax

```
System(command)
```

Arguments

- command - Command to execute

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_TestServer](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_TestServer

TestServer()

Synopsis

Execute Interface Test Server.

Description

Perform test server function and write call report. Results stored in `/var/log/asterisk/testreports/<testid>-server.txt`

Syntax

```
TestServer()
```

Arguments

See Also

- [Asterisk 12 Application_TestClient](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_Transfer

Transfer()

Synopsis

Transfer caller to remote extension.

Description

Requests the remote caller be transferred to a given destination. If TECH (SIP, IAX2, LOCAL etc) is used, only an incoming call with the same channel technology will be transferred. Note that for SIP, if you transfer before call is setup, a 302 redirect SIP message will be returned to the caller.

The result of the application will be reported in the `TRANSFERSTATUS` channel variable:

- `TRANSFERSTATUS`
 - `SUCCESS` - Transfer succeeded.
 - `FAILURE` - Transfer failed.
 - `UNSUPPORTED` - Transfer unsupported by channel driver.

Syntax

```
Transfer(Tech/destination)
```

Arguments

- `dest`
 - `Tech/`
 - `destination`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_TryExec

TryExec()

Synopsis

Executes dialplan application, always returning.

Description

Allows an arbitrary application to be invoked even when not hard coded into the dialplan. To invoke external applications see the application System. Always returns to the dialplan. The channel variable TRYSTATUS will be set to one of:

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

Syntax

```
TryExec (arguments)
```

Arguments

- appname
 - arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_TrySystem

TrySystem()

Synopsis

Try executing a system command.

Description

Executes a command by using system().

Result of execution is returned in the `SYSTEMSTATUS` channel variable:

- `SYSTEMSTATUS`
 - `FAILURE` - Could not execute the specified command.
 - `SUCCESS` - Specified command successfully executed.
 - `APPERROR` - Specified command successfully executed, but returned error code.

Syntax

```
TrySystem(command)
```

Arguments

- `command` - Command to execute

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_UnpauseMonitor

UnpauseMonitor()

Synopsis

Unpause monitoring of a channel.

Description

Unpauses monitoring of a channel on which monitoring had previously been paused with PauseMonitor.

Syntax

```
UnpauseMonitor()
```

Arguments

See Also

- [Asterisk 12 Application_PauseMonitor](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_UnpauseQueueMember

UnpauseQueueMember()

Synopsis

Unpauses a queue member.

Description

Unpauses (resumes calls to) a queue member. This is the counterpart to `PauseQueueMember()` and operates exactly the same way, except it unpauses instead of pausing the given interface.

This application sets the following channel variable upon completion:

- `UPQMSTATUS` - The status of the attempt to unpause a queue member as a text string.
 - `UNPAUSED`
 - `NOTFOUND`
- Example: `UnpauseQueueMember(SIP/3000)`

Syntax

```
UnpauseQueueMember(queueName,interface,options,reason)
```

Arguments

- `queueName`
- `interface`
- `options`
- `reason` - Is used to add extra information to the appropriate `queue_log` entries and manager events.

See Also

- [Asterisk 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_UserEvent

UserEvent()

Synopsis

Send an arbitrary user-defined event to parties interested in a channel (AMI users and relevant res_stasis applications).

Description

Sends an arbitrary event to interested parties, with an optional *body* representing additional arguments. The *body* may be specified as a , delimited list of key:value pairs.

For AMI, each additional argument will be placed on a new line in the event and the format of the event will be:

Event: UserEvent

UserEvent: <specified event name>

[body]

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

For res_stasis applications, the event will be provided as a JSON blob with additional arguments appearing as keys in the object and the *eventname* under the *eventname* key.

Syntax

```
UserEvent ( eventname , body )
```

Arguments

- eventname
- body

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_VoiceMail

VoiceMail()

Synopsis

Leave a Voicemail message.

Description

This application allows the calling party to leave a message for the specified list of mailboxes. When multiple mailboxes are specified, the greeting will be taken from the first mailbox specified. Dialplan execution will stop if the specified mailbox does not exist.

The Voicemail application will exit if any of the following DTMF digits are received:

- 0 - Jump to the 0 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
 - mailbox
 - context
 - mailbox2
 - mailbox
 - context
- options
 - b - Play the busy greeting to the calling party.
 - d - Accept digits for a new extension in context c, if played during the greeting. Context defaults to the current context.
 - c
 - g - Use the specified amount of gain when recording the voicemail message. The units are whole-number decibels (dB). Only works on supported technologies, which is DAHDI only.
 - #
 - s - Skip the playback of instructions for leaving a message to the calling party.
 - u - Play the unavailable greeting.
 - U - Mark message as URGENT.
 - P - Mark message as PRIORITY.

See Also

- [Asterisk 12 Application_VoiceMailMain](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_VoiceMailMain

VoiceMailMain()

Synopsis

Check Voicemail messages.

Description

This application allows the calling party to check voicemail messages. A specific *mailbox*, and optional corresponding *context*, may be specified. If a *mailbox* is not provided, the calling party will be prompted to enter one. If a *context* is not specified, the `default` context will be used.

The VoiceMailMain application will exit if the following DTMF digit is entered as Mailbox or Password, and the extension exists:

- `*` - Jump to the `a` extension in the current dialplan context.

Syntax

```
VoiceMailMain(mailbox@context,options)
```

Arguments

- `mailbox`
 - `mailbox`
 - `context`
- `options`
 - `p` - Consider the *mailbox* parameter as a prefix to the mailbox that is entered by the caller.
 - `g` - Use the specified amount of gain when recording a voicemail message. The units are whole-number decibels (dB).
 - `#`
 - `s` - Skip checking the passcode for the mailbox.
 - `a` - Skip folder prompt and go directly to *folder* specified. Defaults to `INBOX` (or `0`).
 - `folder`
 - `0` - `INBOX`
 - `1` - `Old`
 - `2` - `Work`
 - `3` - `Family`
 - `4` - `Friends`
 - `5` - `Cust1`
 - `6` - `Cust2`
 - `7` - `Cust3`
 - `8` - `Cust4`
 - `9` - `Cust5`

See Also

- [Asterisk 12 Application_VoiceMail](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Application_VoiceMailPlayMsg

VoiceMailPlayMsg()

Synopsis

Play a single voice mail msg from a mailbox by msg id.

Description

This application sets the following channel variable upon completion:

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

Syntax

```
VoiceMailPlayMsg(mailbox@context,msg_id)
```

Arguments

- mailbox
 - mailbox
 - context
- msg_id - The msg id of the msg to play back.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_WaitExten

WaitExten()

Synopsis

Waits for an extension to be entered.

Description

This application waits for the user to enter a new extension for a specified number of *seconds*.



Warning

Use of the application `WaitExten` within a macro will not function as expected. Please use the `Read` application in order to read DTMF from a channel currently executing a macro.

Syntax

```
WaitExten(seconds,options)
```

Arguments

- `seconds` - Can be passed with fractions of a second. For example, `1.5` will ask the application to wait for 1.5 seconds.
- `options`
 - `m` - Provide music on hold to the caller while waiting for an extension.
 - `x` - Specify the class for music on hold. **CHANNEL(musicclass) will be used instead if set**

See Also

- [Asterisk 12 Application_Background](#)
- [Asterisk 12 Function_TIMEOUT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_WaitForSilence](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_WaitForSilence

WaitForSilence()

Synopsis

Waits for a specified amount of silence.

Description

Waits for up to *silencerequired* milliseconds of silence, *iterations* times. An optional *timeout* specified the number of seconds to return after, even if we do not receive the specified amount of silence. Use *timeout* with caution, as it may defeat the purpose of this application, which is to wait indefinitely until silence is detected on the line. This is particularly useful for reverse-911-type call broadcast applications where you need to wait for an answering machine to complete its spiel before playing a message.

Typically you will want to include two or more calls to WaitForSilence when dealing with an answering machine; first waiting for the spiel to finish, then waiting for the beep, etc.

Examples:

WaitForSilence(500,2) will wait for 1/2 second of silence, twice

WaitForSilence(1000) will wait for 1 second of silence, once

WaitForSilence(300,3,10) will wait for 300ms silence, 3 times, and returns after 10 sec, even if silence is not detected

Sets the channel variable WAITSTATUS to one of these values:

- WAITSTATUS
 - SILENCE - if exited with silence detected.
 - TIMEOUT - if exited without silence detected after timeout.

Syntax

```
WaitForSilence(silencerequired,iterations,timeout)
```

Arguments

- *silencerequired*
- *iterations* - If not specified, defaults to 1.
- *timeout* - Is specified only to avoid an infinite loop in cases where silence is never achieved.

See Also

- [Asterisk 12 Application_WaitForNoise](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Application_WaitUntil

WaitUntil()

Synopsis

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

Description

Waits until the given *epoch*.

Sets WAITUNTILSTATUS to one of the following values:

- WAITUNTILSTATUS
 - OK - Wait succeeded.
 - FAILURE - Invalid argument.
 - HANGUP - Channel hungup before time elapsed.
 - PAST - Time specified had already past.

Syntax

```
WaitUntil(epoch)
```

Arguments

- epoch

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_EndWhile](#)
- [Asterisk 12 Application_ExitWhile](#)
- [Asterisk 12 Application_ContinueWhile](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Dialplan Functions

Asterisk 12 Function `_AES_DECRYPT`

`AES_DECRYPT()`

Synopsis

Decrypt a string encoded in base64 with AES given a 16 character key.

Description

Returns the plain text string.

Syntax

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

Arguments

- `key` - AES Key
- `string` - Input string.

See Also

- [Asterisk 12 Function `_AES_ENCRYPT`](#)
- [Asterisk 12 Function `_BASE64_ENCODE`](#)
- [Asterisk 12 Function `_BASE64_DECODE`](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Function_AES_DECRYPT](#)
- [Asterisk 12 Function_BASE64_ENCODE](#)
- [Asterisk 12 Function_BASE64_DECODE](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_AG

AGC()

Synopsis

Apply automatic gain control to audio on a channel.

Description

The AGC function will apply automatic gain control to the audio on the channel that it is executed on. Using `rx` for audio received and `tx` for audio transmitted to the channel. When using this function you set a target audio level. It is primarily intended for use with analog lines, but could be useful for other channels as well. The target volume is set with a number between 1-32768. The larger the number the louder (more gain) the channel will receive.

Examples:

```
exten => 1,1,Set(AGC(rx)=8000)
```

```
exten => 1,2,Set(AGC(tx)=off)
```

Syntax

AGC(channeldirection)

Arguments

- `channeldirection` - This can be either `rx` or `tx`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_AGENT

AGENT()

Synopsis

Gets information about an Agent

Description

Syntax

```
AGENT(AgentId:item)
```

Arguments

- `AgentId`
- `item` - The valid items to retrieve are:
 - `status` - (default) The status of the agent (LOGGEDIN | LOGGEDOUT)
 - `password` - Deprecated. The dialplan handles any agent authentication.
 - `name` - The name of the agent
 - `mohclass` - MusicOnHold class
 - `channel` - The name of the active channel for the Agent (AgentLogin)
 - `fullchannel` - The untruncated name of the active channel for the Agent (AgentLogin)

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `_AMI_CLIENT`

`AMI_CLIENT()`

Synopsis

Checks attributes of manager accounts

Description

Currently, the only supported parameter is "sessions" which will return the current number of active sessions for this AMI account.

Syntax

```
AMI_CLIENT(loginname,field)
```

Arguments

- `loginname` - Login name, specified in `manager.conf`
- `field` - The manager account attribute to return
 - `sessions` - The number of sessions for this AMI account

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_ARRAY

ARRAY()

Synopsis

Allows setting multiple variables at once.

Description

The comma-delimited list passed as a value to which the function is set will be interpreted as a set of values to which the comma-delimited list of variable names in the argument should be set.

Example: Set(ARRAY(var1,var2)=1,2) will set var1 to 1 and var2 to 2

Syntax

```
ARRAY(var1[,var2[,...][,varN]])
```

Arguments

- var1
- var2
- varN

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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
 - MixMonitor
 - Chanspy
 - Volume
 - Speex
 - pitch_shift
 - JACK_HOOK
 - MuteNote that the names are not case-sensitive

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Function_BASE64_ENCODE](#)
- [Asterisk 12 Function_AES_DECRYPT](#)
- [Asterisk 12 Function_AES_ENCRYPT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Function_BASE64_DECODE](#)
- [Asterisk 12 Function_AES_DECRYPT](#)
- [Asterisk 12 Function_AES_ENCRYPT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Function_DB](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CALENDAR_BUSY

CALENDAR_BUSY()

Synopsis

Determine if the calendar is marked busy at this time.

Description

Check the specified calendar's current busy status.

Syntax

```
CALENDAR_BUSY(calendar)
```

Arguments

- calendar

See Also

- [Asterisk 12 Function_CALENDAR_EVENT](#)
- [Asterisk 12 Function_CALENDAR_QUERY](#)
- [Asterisk 12 Function_CALENDAR_QUERY_RESULT](#)
- [Asterisk 12 Function_CALENDAR_WRITE](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CALENDAR_EVENT

CALENDAR_EVENT()

Synopsis

Get calendar event notification data from a notification call.

Description

Whenever a calendar event notification call is made, the event data may be accessed with this function.

Syntax

```
CALENDAR_EVENT(field)
```

Arguments

- `field`
 - `summary` - The VEVENT SUMMARY property or Exchange event 'subject'
 - `description` - The text description of the event
 - `organizer` - The organizer of the event
 - `location` - The location of the event
 - `categories` - The categories of the event
 - `priority` - The priority of the event
 - `calendar` - The name of the calendar associated with the event
 - `uid` - The unique identifier for this event
 - `start` - The start time of the event
 - `end` - The end time of the event
 - `busystate` - The busy state of the event 0=FREE, 1=TENTATIVE, 2=BUSY

See Also

- [Asterisk 12 Function_CALENDAR_BUSY](#)
- [Asterisk 12 Function_CALENDAR_QUERY](#)
- [Asterisk 12 Function_CALENDAR_QUERY_RESULT](#)
- [Asterisk 12 Function_CALENDAR_WRITE](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CALENDAR_QUERY

CALENDAR_QUERY()

Synopsis

Query a calendar server and store the data on a channel

Description

Get a list of events in the currently accessible timeframe of the *calendar* The function returns the id for accessing the result with CALENDAR_QUERY_RESULT()

Syntax

```
CALENDAR_QUERY(calendar[,start[,end]])
```

Arguments

- *calendar* - The calendar that should be queried
- *start* - The start time of the query (in seconds since epoch)
- *end* - The end time of the query (in seconds since epoch)

See Also

- [Asterisk 12 Function_CALENDAR_BUSY](#)
- [Asterisk 12 Function_CALENDAR_EVENT](#)
- [Asterisk 12 Function_CALENDAR_QUERY_RESULT](#)
- [Asterisk 12 Function_CALENDAR_WRITE](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CALENDAR_QUERY_RESULT

CALENDAR_QUERY_RESULT()

Synopsis

Retrieve data from a previously run CALENDAR_QUERY() call

Description

After running CALENDAR_QUERY and getting a result *id*, calling CALENDAR_QUERY with that *id* and a *field* will return the data for that field. If multiple events matched the query, and *entry* is provided, information from that event will be returned.

Syntax

```
CALENDAR_QUERY_RESULT(id,field[,entry])
```

Arguments

- *id* - The query ID returned by CALENDAR_QUERY
- *field*
 - *getnum* - number of events occurring during time range
 - *summary* - A summary of the event
 - *description* - The full event description
 - *organizer* - The event organizer
 - *location* - The event location
 - *categories* - The categories of the event
 - *priority* - The priority of the event
 - *calendar* - The name of the calendar 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 12 Function_CALENDAR_BUSY](#)
- [Asterisk 12 Function_CALENDAR_EVENT](#)
- [Asterisk 12 Function_CALENDAR_QUERY](#)
- [Asterisk 12 Function_CALENDAR_WRITE](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CALENDAR_WRITE

CALENDAR_WRITE()

Synopsis

Write an event to a calendar

Description

Example: CALENDAR_WRITE(calendar,field1,field2,field3)=val1,val2,val3

The field and value arguments can easily be set/passed using the HASHKEYS() and HASH() functions

- CALENDAR_SUCCESS - The status of the write operation to the calendar
 - 1 - The event was successfully written to the calendar.
 - 0 - The event was not written to the calendar due to network issues, permissions, etc.

Syntax

```
CALENDAR_WRITE(calendar,field[,...])
```

Arguments

- calendar - The calendar to write to
- field
 - summary - A summary of the event
 - description - The full event description
 - organizer - The event organizer
 - location - The event location
 - categories - The categories of the event
 - priority - The priority of the event
 - uid - The unique identifier for the event
 - start - The start time of the event (in seconds since epoch)
 - end - The end time of the event (in seconds since epoch)
 - busystate - The busy status of the event 0=FREE, 1=TENTATIVE, 2=BUSY

See Also

- [Asterisk 12 Function_CALENDAR_BUSY](#)
- [Asterisk 12 Function_CALENDAR_EVENT](#)
- [Asterisk 12 Function_CALENDAR_QUERY](#)
- [Asterisk 12 Function_CALENDAR_QUERY_RESULT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_CALLERID

CALLERID()

Synopsis

Gets or sets Caller*ID data on the channel.

Description

Gets or sets Caller*ID data on the channel. Uses channel callerid by default or optional callerid, if specified.

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

- unknown - Unknown
- iso8859-1 - ISO8859-1
- withdrawn - Withdrawn
- iso8859-2 - ISO8859-2
- iso8859-3 - ISO8859-3
- iso8859-4 - ISO8859-4
- iso8859-5 - ISO8859-5
- iso8859-7 - ISO8859-7
- bmp - ISO10646 Bmp String
- utf8 - ISO10646 UTF-8 String

Syntax

```
CALLERID(datatype,CID)
```

Arguments

- datatype - The allowable datatypes are:
 - all
 - name
 - name-valid
 - name-charset
 - name-pres
 - num
 - num-valid
 - num-plan
 - num-pres
 - subaddr
 - subaddr-valid
 - subaddr-type
 - subaddr-odd
 - tag
 - priv-all
 - priv-name
 - priv-name-valid
 - priv-name-charset
 - priv-name-pres
 - priv-num
 - priv-num-valid
 - priv-num-plan
 - priv-num-pres
 - priv-subaddr
 - priv-subaddr-valid
 - priv-subaddr-type
 - priv-subaddr-odd
 - priv-tag
 - ANI-all

- ANI-name
- ANI-name-valid
- ANI-name-charset
- ANI-name-pres
- ANI-num
- ANI-num-valid
- ANI-num-plan
- ANI-num-pres
- ANI-tag
- RDNIS
- DNID
- dnid-num-plan
- dnid-subaddr
- dnid-subaddr-valid
- dnid-subaddr-type
- dnid-subaddr-odd
- CID - Optional Caller*ID to parse instead of using the Caller*ID from the channel. This parameter is only optional when reading the Caller*ID.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CALLERPRES

CALLERPRES()

Synopsis

Gets or sets Caller*ID presentation on the channel.

Description

Gets or sets Caller*ID presentation on the channel. This function is deprecated in favor of CALLERID(num-pres) and CALLERID(name-pres). The following values are valid:

- `allowed_not_screened` - Presentation Allowed, Not Screened.
- `allowed_passed_screen` - Presentation Allowed, Passed Screen.
- `allowed_failed_screen` - Presentation Allowed, Failed Screen.
- `allowed` - Presentation Allowed, Network Number.
- `prohib_not_screened` - Presentation Prohibited, Not Screened.
- `prohib_passed_screen` - Presentation Prohibited, Passed Screen.
- `prohib_failed_screen` - Presentation Prohibited, Failed Screen.
- `prohib` - Presentation Prohibited, Network Number.
- `unavailable` - Number Unavailable.

Syntax

```
CALLERPRES()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CDR

CDR()

Synopsis

Gets or sets a CDR variable.

Description

All of the CDR field names are read-only, except for `accountcode`, `userfield`, and `amaflags`. You may, however, supply a name not on the above list, and create your own variable, whose value can be changed with this function, and this variable will be stored on the CDR.



Note

CDRs can only be modified before the bridge between two channels is torn down. For example, CDRs may not be modified after the `Dial` application has returned.

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

Syntax

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

Arguments

- `name` - CDR field name:
 - `clid` - Caller ID.
 - `lastdata` - Last application arguments.
 - `disposition` - The final state of the CDR.
 - 0 - NO ANSWER
 - 1 - NO ANSWER (NULL record)
 - 2 - FAILED
 - 4 - BUSY
 - 8 - ANSWERED
 - 16 - CONGESTION
 - `src` - Source.
 - `start` - Time the call started.
 - `amaflags` - R/W the Automatic Message Accounting (AMA) flags on the channel. When read from a channel, the integer value will always be returned. When written to a channel, both the string format or integer value is accepted.
 - 1 - OMIT
 - 2 - BILLING
 - 3 - DOCUMENTATION



Warning

Accessing this setting is deprecated in CDR. Please use the `CHANNEL` function instead.

- `dst` - Destination.
- `answer` - Time the call was answered.
- `accountcode` - The channel's account code.



Warning

Accessing this setting is deprecated in CDR. Please use the `CHANNEL` function instead.

- `dcontext` - Destination context.
- `end` - Time the call ended.
- `uniqueid` - The channel's unique id.
- `dstchannel` - Destination channel.
- `duration` - Duration of the call.
- `userfield` - The channel's user specified field.

- `lastapp` - Last application.
- `billsec` - Duration of the call once it was answered.
- `channel` - Channel name.
- `sequence` - CDR sequence number.
- `options`
 - `f` - Returns `billsec` or duration fields as floating point values.
 - `u` - Retrieves the raw, unprocessed value.
For example, 'start', 'answer', and 'end' will be retrieved as epoch values, when the `u` option is passed, but formatted as YYYY-MM-DD HH:MM:SS otherwise. Similarly, `disposition` and `amaflags` will return their raw integral values.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CDR_PROP

CDR_PROP()

Synopsis

Set a property on a channel's CDR.

Description

This function sets a property on a channel's CDR. Properties alter the behavior of how the CDR operates for that channel.

Syntax

```
CDR_PROP(name)
```

Arguments

- `name` - The property to set on the CDR.
 - `party_a` - Set this channel as the preferred Party A when channels are associated together.
Write-Only
 - `disable` - Disable CDRs for this channel.
Write-Only

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CHANNEL

CHANNEL()

Synopsis

Gets/sets various pieces of information about the channel.

Description

Gets/sets various pieces of information about the channel, additional *item* may be available from the channel driver; see its documentation for details. Any *item* requested that is not available on the current channel will return an empty string.

Syntax

```
CHANNEL(item)
```

Arguments

- *item* - Standard items (provided by all channel technologies) are:
 - *amaflags* - R/W the Automatic Message Accounting (AMA) flags on the channel. When read from a channel, the integer value will always be returned. When written to a channel, both the string format or integer value is accepted.
 - 1 - OMIT
 - 2 - BILLING
 - 3 - DOCUMENTATION
 - *accountcode* - R/W the channel's account code.
 - *audioreadformat* - R/O format currently being read.
 - *audionativeformat* - R/O format used natively for audio.
 - *audiowriteformat* - R/O format currently being written.
 - *dtmf_features* - R/W The channel's DTMF bridge features. May include one or more of 'T' 'K' 'H' 'W' and 'X' in a similar manner to options in the `Dial` application. When setting it, the features string must be all upper case.
 - *callgroup* - R/W numeric call pickup groups that this channel is a member.
 - *pickupgroup* - R/W numeric call pickup groups this channel can pickup.
 - *namedcallgroup* - R/W named call pickup groups that this channel is a member.
 - *namedpickupgroup* - R/W named call pickup groups this channel can pickup.
 - *channeltype* - R/O technology used for channel.
 - *checkhangup* - R/O Whether the channel is hanging up (1/0)
 - *after_bridge_goto* - R/W the parseable goto string indicating where the channel is expected to return to in the PBX after exiting the next bridge it joins on the condition that it doesn't hang up. The parseable goto string uses the same syntax as the `Goto` application.
 - *hangup_handler_pop* - W/O Replace the most recently added hangup handler with a new hangup handler on the channel if supplied. The assigned string is passed to the `Gosub` application when the channel is hung up. Any optionally omitted context and exten are supplied by the channel pushing the handler before it is pushed.
 - *hangup_handler_push* - W/O Push a hangup handler onto the channel hangup handler stack. The assigned string is passed to the `Gosub` application when the channel is hung up. Any optionally omitted context and exten are supplied by the channel pushing the handler before it is pushed.
 - *hangup_handler_wipe* - W/O Wipe the entire hangup handler stack and replace with a new hangup handler on the channel if supplied. The assigned string is passed to the `Gosub` application when the channel is hung up. Any optionally omitted context and exten are supplied by the channel pushing the handler before it is pushed.
 - *language* - R/W language for sounds played.
 - *musicclass* - R/W class (from `musiconhold.conf`) for hold music.
 - *name* - The name of the channel
 - *parkinglot* - R/W parkinglot for parking.
 - *rxgain* - R/W set rxgain level on channel drivers that support it.
 - *secure_bridge_signaling* - Whether or not channels bridged to this channel require secure signaling
 - *secure_bridge_media* - Whether or not channels bridged to this channel require secure media
 - *state* - R/O state for channel
 - *tonezone* - R/W zone for indications played
 - *transfercapability* - R/W ISDN Transfer Capability, one of:

- SPEECH
- DIGITAL
- RESTRICTED_DIGITAL
- 3K1AUDIO
- DIGITAL_W_TONES
- VIDEO
- txgain - R/W set txgain level on channel drivers that support it.
- videonativeformat - R/O format used natively for video
- trace - R/W whether or not context tracing is enabled, only available if **CHANNEL_TRACE** is defined.
chan_sip provides the following additional options:
 - peerip - R/O Get the IP address of the peer.
 - recvip - R/O Get the source IP address of the peer.
 - from - R/O Get the URI from the From: header.
 - uri - R/O Get the URI from the Contact: header.
 - useragent - R/O Get the useragent.
 - peername - R/O Get the name of the peer.
 - t38passthrough - R/O 1 if T38 is offered or enabled in this channel, otherwise 0
 - rtpqos - R/O Get QOS information about the RTP stream
This option takes two additional arguments:

Argument 1:

 - audio Get data about the audio stream
 - video Get data about the video stream
 - text Get data about the text stream

Argument 2:

 - local_ssrc Local SSRC (stream ID)
 - local_lostpackets Local lost packets
 - local_jitter Local calculated jitter
 - local_maxjitter Local calculated jitter (maximum)
 - local_minjitter Local calculated jitter (minimum)
 - {{local_normdevjitter}} Local calculated jitter (normal deviation)
 - local_stdevjitter Local calculated jitter (standard deviation)
 - local_count Number of received packets
 - remote_ssrc Remote SSRC (stream ID)
 - {{remote_lostpackets}} Remote lost packets
 - remote_jitter Remote reported jitter
 - remote_maxjitter Remote calculated jitter (maximum)
 - remote_minjitter Remote calculated jitter (minimum)
 - {{remote_normdevjitter}} Remote calculated jitter (normal deviation)
 - {{remote_stdevjitter}} Remote calculated jitter (standard deviation)
 - remote_count Number of transmitted packets
 - rtt Round trip time
 - maxrtt Round trip time (maximum)
 - minrtt Round trip time (minimum)
 - normdevrtt Round trip time (normal deviation)
 - stdevrtt Round trip time (standard deviation)
 - all All statistics (in a form suited to logging, but not for parsing)
- rtpdest - R/O Get remote RTP destination information.
This option takes one additional argument:

Argument 1:

 - audio Get audio destination
 - video Get video destination
 - text Get text destination

Defaults to audio if unspecified.
- rtpsource - R/O Get source RTP destination information.
This option takes one additional argument:

Argument 1:

 - audio Get audio destination
 - video Get video destination

text Get text destination
Defaults to audio if unspecified.

Technology: PJSIP

rtp - R/O Retrieve media related information.

- **type** - When *rtp* is specified, the *type* parameter must be provided. It specifies which RTP parameter to read.
 - **src** - Retrieve the local address for RTP.
 - **dest** - Retrieve the remote address for RTP.
 - **direct** - If direct media is enabled, this address is the remote address used for RTP.
 - **secure** - Whether or not the media stream is encrypted.
 - 0 - The media stream is not encrypted.
 - 1 - The media stream is encrypted.
 - **hold** - Whether or not the media stream is currently restricted due to a call hold.
 - 0 - The media stream is not held.
 - 1 - The media stream is held.
- **media_type** - When *rtp* is specified, the *media_type* parameter may be provided. It specifies which media stream the chosen RTP parameter should be retrieved from.
 - **audio** - Retrieve information from the audio media stream.



Note

If not specified, **audio** is used by default.

- **video** - Retrieve information from the video media stream.
- **rtcp** - R/O Retrieve RTCP statistics.
- **statistic** - When *rtcp* is specified, the *statistic* parameter must be provided. It specifies which RTCP statistic parameter to read.
 - **all** - Retrieve a summary of all RTCP statistics.

The following data items are returned in a semi-colon delineated list:

 - **ssrc** - Our Synchronization Source identifier
 - **themssrc** - Their Synchronization Source identifier
 - **lp** - Our lost packet count
 - **rxjitter** - Received packet jitter
 - **rxcount** - Received packet count
 - **txjitter** - Transmitted packet jitter
 - **txcount** - Transmitted packet count
 - **rlp** - Remote lost packet count
 - **rtt** - Round trip time
 - **all_jitter** - Retrieve a summary of all RTCP Jitter statistics.

The following data items are returned in a semi-colon delineated list:

 - **minrxjitter** - Our minimum jitter
 - **maxrxjitter** - Our max jitter
 - **avgrxjitter** - Our average jitter
 - **stdevrxjitter** - Our jitter standard deviation
 - **reported_minjitter** - Their minimum jitter
 - **reported_maxjitter** - Their max jitter
 - **reported_avgjitter** - Their average jitter
 - **reported_stdevjitter** - Their jitter standard deviation
 - **all_loss** - Retrieve a summary of all RTCP packet loss statistics.

The following data items are returned in a semi-colon delineated list:

 - **minrxlost** - Our minimum lost packets
 - **maxrxlost** - Our max lost packets
 - **avgrxlost** - Our average lost packets
 - **stdevrxlost** - Our lost packets standard deviation
 - **reported_minlost** - Their minimum lost packets
 - **reported_maxlost** - Their max lost packets
 - **reported_avglost** - Their average lost packets
 - **reported_stdevlost** - Their lost packets standard deviation
 - **all_rtt** - Retrieve a summary of all RTCP round trip time information.

The following data items are returned in a semi-colon delineated list:

- `minrtt` - Minimum round trip time
- `maxrtt` - Maximum round trip time
- `avgrtt` - Average round trip time
- `stdevrtt` - Standard deviation round trip time
- `txcount` - Transmitted packet count
- `rxcount` - Received packet count
- `txjitter` - Transmitted packet jitter
- `rxjitter` - Received packet jitter
- `remote_maxjitter` - Their max jitter
- `remote_minjitter` - Their minimum jitter
- `remote_normdevjitter` - Their average jitter
- `remote_stdevjitter` - Their jitter standard deviation
- `local_maxjitter` - Our max jitter
- `local_minjitter` - Our minimum jitter
- `local_normdevjitter` - Our average jitter
- `local_stdevjitter` - Our jitter standard deviation
- `txploss` - Transmitted packet loss
- `rxploss` - Received packet loss
- `remote_maxrxploss` - Their max lost packets
- `remote_minrxploss` - Their minimum lost packets
- `remote_normdevrxploss` - Their average lost packets
- `remote_stdevrxploss` - Their lost packets standard deviation
- `local_maxrxploss` - Our max lost packets
- `local_minrxploss` - Our minimum lost packets
- `local_normdevrxploss` - Our average lost packets
- `local_stdevrxploss` - Our lost packets standard deviation
- `rtt` - Round trip time
- `maxrtt` - Maximum round trip time
- `minrtt` - Minimum round trip time
- `normdevrtt` - Average round trip time
- `stdevrtt` - Standard deviation round trip time
- `local_ssrc` - Our Synchronization Source identifier
- `remote_ssrc` - Their Synchronization Source identifier
- `media_type` - When `rtcp` is specified, the `media_type` parameter may be provided. It specifies which media stream the chosen RTCP parameter should be retrieved from.
 - `audio` - Retrieve information from the audio media stream.



Note

If not specified, `audio` is used by default.

- `video` - Retrieve information from the video media stream.
- `endpoint` - R/O The name of the endpoint associated with this channel. Use the `PJSIP_ENDPOINT` function to obtain further endpoint related information.
- `pjsip` - R/O Obtain information about the current PJSIP channel and its session.
- `type` - When `pjsip` is specified, the `type` parameter must be provided. It specifies which signalling parameter to read.
 - `secure` - Whether or not the signalling uses a secure transport.
 - 0 - The signalling uses a non-secure transport.
 - 1 - The signalling uses a secure transport.
 - `target_uri` - The request URI of the `INVITE` request associated with the creation of this channel.
 - `local_uri` - The local URI.
 - `remote_uri` - The remote URI.
 - `t38state` - The current state of any T.38 fax on this channel.
 - `DISABLED` - T.38 faxing is disabled on this channel.
 - `LOCAL_REINVITE` - Asterisk has sent a `re-INVITE` to the remote end to initiate a T.38 fax.
 - `REMOTE_REINVITE` - The remote end has sent a `re-INVITE` to Asterisk to initiate a T.38 fax.
 - `ENABLED` - A T.38 fax session has been enabled.
 - `REJECTED` - A T.38 fax session was attempted but was rejected.

- `local_addr` - On inbound calls, the full IP address and port number that the `INVITE` request was received on. On outbound calls, the full IP address and port number that the `INVITE` request was transmitted from.
 - `remote_addr` - On inbound calls, the full IP address and port number that the `INVITE` request was received from. On outbound calls, the full IP address and port number that the `INVITE` request was transmitted to.
- `chan_iax2` provides the following additional options:
- `osptoken` - R/O Get the peer's osptoken.
 - `peerip` - R/O Get the peer's ip address.
 - `peername` - R/O Get the peer's username.
 - `secure_signaling` - R/O Get the if the IAX channel is secured.
 - `secure_media` - R/O Get the if the IAX channel is secured.
- `chan_dahdi` provides the following additional options:
- `dahdi_channel` - R/O DAHDI channel related to this channel.
 - `dahdi_span` - R/O DAHDI span related to this channel.
 - `dahdi_type` - R/O DAHDI channel type, one of:
 - `analog`
 - `mfc/r2`
 - `pri`
 - `pseudo`
 - `ss7`
 - `keypad_digits` - R/O PRI Keypad digits that came in with the SETUP message.
 - `reversecharge` - R/O PRI Reverse Charging Indication, one of:
 - `-1` - None
 - `{{ 1 }}` - Reverse Charging Requested
 - `no_media_path` - R/O PRI Nonzero if the channel has no B channel. The channel is either on hold or a call waiting call.
 - `buffers` - W/O Change the channel's buffer policy (for the current call only)
This option takes two arguments:
Number of buffers,
Buffer policy being one of:
`full`
`immediate`
`half`
 - `echocan_mode` - W/O Change the configuration of the active echo canceller on the channel (if any), for the current call only.
Possible values are:
`{{on}}`Normal mode (the echo canceller is actually reinitialized)
`{{off}}`Disabled
`{{fax}}`FAX/data mode (NLP disabled if possible, otherwise completely disabled)
`{{voice}}`Voice mode (returns from FAX mode, reverting the changes that were made)
- `chan_oo323` provides the following additional options:
- `faxdetect` - R/W Fax Detect
Returns 0 or 1
Write yes or no
 - `t38support` - R/W t38support
Returns 0 or 1
Write yes or no
 - `h323id_url` - R/O Returns caller URL
 - `caller_h323id` - R/O Returns caller h323id
 - `caller_dialedigits` - R/O Returns caller dialed digits
 - `caller_email` - R/O Returns caller email
 - `callee_email` - R/O Returns callee email
 - `callee_dialedigits` - R/O Returns callee dialed digits
 - `caller_url` - R/O Returns caller URL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403680

Asterisk 12 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 Unknown

Asterisk 12 Function_CHECKSIPDOMAIN

CHECKSIPDOMAIN()

Synopsis

Checks if domain is a local domain.

Description

This function checks if the *domain* in the argument is configured as a local SIP domain that this Asterisk server is configured to handle. Returns the domain name if it is locally handled, otherwise an empty string. Check the `domain=` configuration in `sip.conf`.

Syntax

```
CHECKSIPDOMAIN(domain)
```

Arguments

- `domain`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CONFBRIDGE

CONFBRIDGE()

Synopsis

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

Description

---- Example 1 ----

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

```
exten => 1,1,Answer()
```

```
exten => 1,n,Set(CONFBRIDGE(user,announce_join_leave)=yes)
```

```
exten => 1,n,Set(CONFBRIDGE(user,startmuted)=yes)
```

```
exten => 1,n,ConfBridge(1)
```

---- Example 2 ----

This example shows how to use a predefined user or bridge profile in `confbridge.conf` as a template for a dynamic profile. Here we make a admin/marked user out of the `default_user` profile that is already defined in `confbridge.conf`.

```
exten => 1,1,Answer()
```

```
exten => 1,n,Set(CONFBRIDGE(user,template)=default_user)
```

```
exten => 1,n,Set(CONFBRIDGE(user,admin)=yes)
```

```
exten => 1,n,Set(CONFBRIDGE(user,marked)=yes)
```

```
exten => 1,n,ConfBridge(1)
```

Syntax

`CONFBRIDGE(type,option)`

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `_CONFBRIDGE_INFO`

`CONFBRIDGE_INFO()`

Synopsis

Get information about a ConfBridge conference.

Description

This function returns a non-negative integer for valid conference identifiers (0 or 1 for `locked`) and "" for invalid conference identifiers.

Syntax

```
CONFBRIDGE_INFO(type,conf)
```

Arguments

- `type` - Type can be `parties`, `admins`, `marked`, or `locked`.
- `conf` - Conf refers to the name of the conference being referenced.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `_CONNECTEDLINE`

`CONNECTEDLINE()`

Synopsis

Gets or sets Connected Line data on the channel.

Description

Gets or sets Connected Line data on the channel.

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

- `unknown` - Unknown
- `iso8859-1` - ISO8859-1
- `withdrawn` - Withdrawn
- `iso8859-2` - ISO8859-2
- `iso8859-3` - ISO8859-3
- `iso8859-4` - ISO8859-4
- `iso8859-5` - ISO8859-5
- `iso8859-7` - ISO8859-7
- `bmp` - ISO10646 Bmp String
- `utf8` - ISO10646 UTF-8 String

Syntax

```
CONNECTEDLINE(datatype,i)
```

Arguments

- `datatype` - The allowable datatypes are:
 - `all`
 - `name`
 - `name-valid`
 - `name-charset`
 - `name-pres`
 - `num`
 - `num-valid`
 - `num-plan`
 - `num-pres`
 - `subaddr`
 - `subaddr-valid`
 - `subaddr-type`
 - `subaddr-odd`
 - `tag`
 - `priv-all`
 - `priv-name`
 - `priv-name-valid`
 - `priv-name-charset`
 - `priv-name-pres`
 - `priv-num`
 - `priv-num-valid`
 - `priv-num-plan`
 - `priv-num-pres`
 - `priv-subaddr`
 - `priv-subaddr-valid`
 - `priv-subaddr-type`
 - `priv-subaddr-odd`
 - `priv-tag`

- `i` - If set, this will prevent the channel from sending out protocol messages because of the value being set

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CSV_QUOTE

CSV_QUOTE()

Synopsis

Quotes a given string for use in a CSV file, escaping embedded quotes as necessary

Description

Example: \${CSV_QUOTE("a,b" 123)} will return ""a,b"" 123"

Syntax

```
CSV_QUOTE(string)
```

Arguments

- `string`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Function_CURLOPT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CURLOPT

CURLOPT()

Synopsis

Sets various options for future invocations of CURL.

Description

Options may be set globally or per channel. Per-channel settings will override global settings.

Syntax

CURLOPT(*key*)

Arguments

- *key*
 - *cookie* - A cookie to send with the request. Multiple cookies are supported.
 - *conntimeout* - Number of seconds to wait for a connection to succeed
 - *dnstimeout* - Number of seconds to wait for DNS to be resolved
 - *ftptext* - For FTP URIs, force a text transfer (boolean)
 - *ftptimeout* - For FTP URIs, number of seconds to wait for a server response
 - *header* - Include header information in the result (boolean)
 - *httptimeout* - For HTTP(S) URIs, number of seconds to wait for a server response
 - *maxredirs* - Maximum number of redirects to follow
 - *proxy* - Hostname or IP address to use as a proxy server
 - *proxytype* - Type of proxy
 - *http*
 - *socks4*
 - *socks5*
 - *proxyport* - Port number of the proxy
 - *proxyuserpwd* - A *username:password* combination to use for authenticating requests through a proxy
 - *referer* - Referer URL to use for the request
 - *useragent* - UserAgent string to use for the request
 - *userpwd* - A *username:password* to use for authentication when the server response to an initial request indicates a 401 status code.
 - *ssl_verifypeer* - Whether to verify the server certificate against a list of known root certificate authorities (boolean).
 - *hashcompat* - Assuming the responses will be in *key1=value1&key2=value2* format, reformat the response such that it can be used by the *HASH* function.
 - *yes*
 - *no*
 - *legacy* - Also translate + to the space character, in violation of current RFC standards.

See Also

- [Asterisk 12 Function_CURL](#)
- [Asterisk 12 Function_HASH](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_CUT

CUT()

Synopsis

Slices and dices strings, based upon a named delimiter.

Description

Cut out information from a string (*varname*), based upon a named delimiter.

Syntax

```
CUT(varname,char-delim,range-spec)
```

Arguments

- *varname* - Variable you want cut
- *char-delim* - Delimiter, defaults to -
- *range-spec* - Number of the field you want (1-based offset), may also be specified as a range (with -) or group of ranges and fields (with &)

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_DBdel](#)
- [Asterisk 12 Function_DB_DELETE](#)
- [Asterisk 12 Application_Dbdeltree](#)
- [Asterisk 12 Function_DB_EXISTS](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_DB_DELETE

DB_DELETE()

Synopsis

Return a value from the database and delete it.

Description

This function will retrieve a value from the Asterisk database and then remove that key from the database. `DB_RESULT` will be set to the key's value if it exists.



Note

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be read from the dialplan, and not directly from external protocols. It can, however, be executed as a write operation (`DB_DELETE(family, key)=ignored`)

Syntax

```
DB_DELETE(family/key)
```

Arguments

- `family`
- `key`

See Also

- [Asterisk 12 Application_DBdel](#)
- [Asterisk 12 Function_DB](#)
- [Asterisk 12 Application_DBdeltree](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 Function_DB_EXISTS

DB_EXISTS()

Synopsis

Check to see if a key exists in the Asterisk database.

Description

This function will check to see if a key exists in the Asterisk database. If it exists, the function will return 1. If not, it will return 0. Checking for existence of a database key will also set the variable DB_RESULT to the key's value if it exists.

Syntax

```
DB_EXISTS( family/key)
```

Arguments

- family
- key

See Also

- [Asterisk 12 Function_DB](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_DB_KEYS

DB_KEYS()

Synopsis

Obtain a list of keys within the Asterisk database.

Description

This function will return a comma-separated list of keys existing at the prefix specified within the Asterisk database. If no argument is provided, then a list of key families will be returned.

Syntax

```
DB_KEYS(prefix)
```

Arguments

- `prefix`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_DEC

DEC()

Synopsis

Decrements the value of a variable, while returning the updated value to the dialplan

Description

Decrements the value of a variable, while returning the updated value to the dialplan

Example: DEC(MyVAR) - Decrements MyVar

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

Syntax

```
DEC(variable)
```

Arguments

- `variable` - The variable name to be manipulated, without the braces.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function `_DEVICE_STATE`

`DEVICE_STATE()`

Synopsis

Get or Set a device state.

Description

The `DEVICE_STATE` function can be used to retrieve the device state from any device state provider. For example:

```
NoOp(SIP/mypeer has state ${DEVICE_STATE(SIP/mypeer)})
```

```
NoOp(Conference number 1234 has state ${DEVICE_STATE(MeetMe:1234)})
```

The `DEVICE_STATE` function can also be used to set custom device state from the dialplan. The `Custom:` prefix must be used. For example:

```
Set(DEVICE_STATE(Custom:lamp1)=BUSY)
```

```
Set(DEVICE_STATE(Custom:lamp2)=NOT_INUSE)
```

You can subscribe to the status of a custom device state using a hint in the dialplan:

```
exten => 1234,hint,Custom:lamp1
```

The possible values for both uses of this function are:

UNKNOWN | NOT_INUSE | INUSE | BUSY | INVALID | UNAVAILABLE | RINGING | RINGINUSE | ONHOLD

Syntax

```
DEVICE_STATE(device)
```

Arguments

- device

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function `_DIALPLAN_EXISTS`

`DIALPLAN_EXISTS()`

Synopsis

Checks the existence of a dialplan target.

Description

This function returns 1 if the target exists. Otherwise, it returns 0.

Syntax

```
DIALPLAN_EXISTS(context,extension,priority)
```

Arguments

- context
- extension
- priority

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_DUNDILOOKUP

DUNDILOOKUP()

Synopsis

Do a DUNDi lookup of a phone number.

Description

This will do a DUNDi lookup of the given phone number.

This function will return the Technology/Resource found in the first result in the DUNDi lookup. If no results were found, the result will be blank.

Syntax

```
DUNDILOOKUP(number,context,options)
```

Arguments

- number
- context - If not specified the default will be e164.
- options
 - b - Bypass the internal DUNDi cache

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_ENUMLOOKUP

ENUMLOOKUP()

Synopsis

General or specific querying of NAPTR records for ENUM or ENUM-like DNS pointers.

Description

For more information see `doc/AST.pdf`.

Syntax

```
ENUMLOOKUP(number,method-type,options,record#,zone-suffix)
```

Arguments

- `number`
- `method-type` - If no *method-type* is given, the default will be `sip`.
- `options`
 - `c` - Returns an integer count of the number of NAPTRs of a certain RR type.
Combination of `c` and Method-type of `ALL` will return a count of all NAPTRs for the record or -1 on error.
 - `u` - Returns the full URI and does not strip off the URI-scheme.
 - `s` - Triggers ISN specific rewriting.
 - `i` - Looks for branches into an Infrastructure ENUM tree.
 - `d` - for a direct DNS lookup without any flipping of digits.
- `record#` - If no *record#* is given, defaults to 1.
- `zone-suffix` - If no *zone-suffix* is given, the default will be `e164.arpa`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_ENUMRESULT

ENUMRESULT()

Synopsis

Retrieve results from a ENUMQUERY.

Description

This function will retrieve results from a previous use of the ENUMQUERY function.

Syntax

```
ENUMRESULT(id,resultnum)
```

Arguments

- `id` - The identifier returned by the ENUMQUERY function.
- `resultnum` - The number of the result that you want to retrieve.
Results start at 1. If this argument is specified as `getnum`, then it will return the total number of results that are available or -1 on error.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_ENV

ENV()

Synopsis

Gets or sets the environment variable specified.

Description

Variables starting with `AST_` are reserved to the system and may not be set.

Syntax

```
ENV(varname)
```

Arguments

- `varname` - Environment variable name

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_EVAL

EVAL()

Synopsis

Evaluate stored variables

Description

Using EVAL basically causes a string to be evaluated twice. When a variable or expression is in the dialplan, it will be evaluated at runtime. However, if the results of the evaluation is in fact another variable or expression, using EVAL will have it evaluated a second time.

Example: If the MYVAR contains OTHERVAR, then the result of \${EVAL(MYVAR)} in the dialplan will be the contents of OTHERVAR. Normally just putting MYVAR in the dialplan the result would be OTHERVAR.

Syntax

```
EVAL(variable)
```

Arguments

- variable

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Application_RaiseException](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_EXISTS

EXISTS()

Synopsis

Test the existence of a value.

Description

Returns 1 if exists, 0 otherwise.

Syntax

```
EXISTS(data)
```

Arguments

- data

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_FAXOPT_res_fax

FAXOPT() - [res_fax]

Synopsis

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

Description

FAXOPT can be used to override the settings for a FAX session listed in `res_fax.conf`, it can also be used to 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)
 - `faxdetect` - R/W Enable FAX detect with optional timeout in seconds (yes,t38,cng[,timeout]/no)
 - `pages` - R/O Number of pages transferred.
 - `rate` - R/O Negotiated transmission rate.
 - `remotestationid` - R/O Remote Station Identification after transmission.
 - `resolution` - R/O Negotiated image resolution after transmission.
 - `sessionid` - R/O Session ID of the FAX transmission.
 - `status` - R/O Result Status of the FAX transmission.
 - `statusstr` - R/O Verbose Result Status of the FAX transmission.

See Also

- [Asterisk 12 Application_ReceiveFax](#)
- [Asterisk 12 Application_SendFax](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_FEATURE

FEATURE()

Synopsis

Get or set a feature option on a channel.

Description

When this function is used as a read, it will get the current value of the specified feature option for this channel. It will be the value of this option configured in features.conf if a channel specific value has not been set. This function can also be used to set a channel specific value for the supported feature options.

Syntax

```
FEATURE(option_name)
```

Arguments

- `option_name` - The allowed values are:
 - `inherit` - Inherit feature settings made in FEATURE or FEATUREMAP to child channels.
 - `featuredigittimeout` - Milliseconds allowed between digit presses when entering a feature code.
 - `transferdigittimeout` - Seconds allowed between digit presses when dialing a transfer destination
 - `atxfernoanswertimeout` - Seconds to wait for attended transfer destination to answer
 - `atxferdropcall` - Hang up the call entirely if the attended transfer fails
 - `atxferloopdelay` - Seconds to wait between attempts to re-dial transfer destination
 - `atxfercallbackretries` - Number of times to re-attempt dialing a transfer destination
 - `xfersound` - Sound to play to during transfer and transfer-like operations.
 - `xferfailsound` - Sound to play to a transferee when a transfer fails
 - `atxferabort` - Digits to dial to abort an attended transfer attempt
 - `atxfercomplete` - Digits to dial to complete an attended transfer
 - `atxferthreeway` - Digits to dial to change an attended transfer into a three-way call
 - `pickupexten` - Digits used for picking up ringing calls
 - `pickupsound` - Sound to play to picker when a call is picked up
 - `pickupfailsound` - Sound to play to picker when a call cannot be picked up
 - `courtesytone` - Sound to play when automon or automixmon is activated
 - `recordingfailsound` - Sound to play when automon or automixmon is attempted but fails to start

See Also

- [Asterisk 12 Function_FEATUREMAP](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402154

Asterisk 12 Function_FEATUREMAP

FEATUREMAP()

Synopsis

Get or set a feature map to a given value on a specific channel.

Description

When this function is used as a read, it will get the current digit sequence mapped to the specified feature for this channel. This value will be the one configured in features.conf if a channel specific value has not been set. This function can also be used to set a channel specific value for a feature mapping.

Syntax

```
FEATUREMAP(feature_name)
```

Arguments

- `feature_name` - The allowed values are:
 - `atxfer` - Attended Transfer
 - `blindxfer` - Blind Transfer
 - `automon` - Auto Monitor
 - `disconnect` - Call Disconnect
 - `parkcall` - Park Call
 - `automixmon` - Auto MixMonitor

See Also

- [Asterisk 12 Function_FEATURE](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_FILE

FILE()

Synopsis

Read or write text file.

Description

Read and write text file in character and line mode.

Examples:

Read mode (byte):

;reads the entire content of the file.

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

;reads from the 11th byte to the end of the file (i.e. skips the first 10).

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

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

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

Read mode (line):

; reads the 3rd line of the file.

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

; reads the 3rd and 4th lines of the file.

```
Set(foo=${FILE(/tmp/test.txt,3,2,l)})
```

; reads from the third line to the end of the file.

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

; reads the last three lines of the file.

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

; reads the 3rd line of a DOS-formatted file.

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

Write mode (byte):

; truncate the file and write "bar"

```
Set(FILE(/tmp/test.txt)=bar)
```

; Append "bar"

```
Set(FILE(/tmp/test.txt,,a)=bar)
```

; Replace the first byte with "bar" (replaces 1 character with 3)

```
Set(FILE(/tmp/test.txt,0,1)=bar)
```

; Replace 10 bytes beginning at the 21st byte of the file with "bar"

```
Set(FILE(/tmp/test.txt,20,10)=bar)
```

; Replace all bytes from the 21st with "bar"

```
Set(FILE(/tmp/test.txt,20)=bar)
```

; Insert "bar" after the 4th character

```
Set(FILE(/tmp/test.txt,4,0)=bar)
```

Write mode (line):

```
; Replace the first line of the file with "bar"
```

```
Set(FILE(/tmp/foo.txt,0,1,l)=bar)
```

```
; Replace the last line of the file with "bar"
```

```
Set(FILE(/tmp/foo.txt,-1,,l)=bar)
```

```
; Append "bar" to the file with a newline
```

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



Note

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
FILE(filename,offset,length,options,format)
```

Arguments

- `filename`
- `offset` - Maybe specified as any number. If negative, *offset* specifies the number of bytes back from the end of the file.
- `length` - If specified, will limit the length of the data read to that size. If negative, trims *length* bytes from the end of the file.
- `options`
 - `l` - Line mode: offset and length are assumed to be measured in lines, instead of byte offsets.
 - `a` - In write mode only, the append option is used to append to the end of the file, instead of overwriting the existing file.
 - `d` - In write mode and line mode only, this option does not automatically append a newline string to the end of a value. This is useful for deleting lines, instead of setting them to blank.
- `format` - The *format* parameter may be used to delimit the type of line terminators in line mode.
 - `u` - Unix newline format.
 - `d` - DOS newline format.
 - `m` - Macintosh newline format.

See Also

- [Asterisk 12 Function_FILE_COUNT_LINE](#)
- [Asterisk 12 Function_FILE_FORMAT](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 Function_FILE_COUNT_LINE

FILE_COUNT_LINE()

Synopsis

Obtains the number of lines of a text file.

Description

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



Note

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
FILE_COUNT_LINE(filename,format)
```

Arguments

- `filename`
- `format` - Format may be one of the following:
 - `u` - Unix newline format.
 - `d` - DOS newline format.
 - `m` - Macintosh newline format.



Note

If not specified, an attempt will be made to determine the newline format type.

See Also

- [Asterisk 12 Function_FILE](#)
- [Asterisk 12 Function_FILE_FORMAT](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 Function_FILE_FORMAT

FILE_FORMAT()

Synopsis

Return the newline format of a text file.

Description

Return the line terminator type:

'u' - Unix "\n" format

'd' - DOS "\r\n" format

'm' - Macintosh "\r" format

'x' - Cannot be determined



Note

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
FILE_FORMAT(filename)
```

Arguments

- `filename`

See Also

- [Asterisk 12 Function_FILE](#)
- [Asterisk 12 Function_FILE_COUNT_LINE](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 Function_FILTER

FILTER()

Synopsis

Filter the string to include only the allowed characters

Description

Permits all characters listed in *allowed-chars*, filtering all others outs. In addition to literally listing the characters, you may also use ranges of characters (delimited by a -

Hexadecimal characters started with a \x(i.e. \x20)

Octal characters started with a \0 (i.e. \040)

Also \t,\n and \r are recognized.



Note

If you want the - character it needs to be prefixed with a {}

Syntax

```
FILTER(allowed-chars,string)
```

Arguments

- `allowed-chars`
- `string`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_FRAME_TRACE

FRAME_TRACE()

Synopsis

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

Description

Examples:

exten => 1,1,Set(FRAME_TRACE(white)=DTMF_BEGIN,DTMF_END); view only DTMF frames.

exten => 1,1,Set(FRAME_TRACE(white)=DTMF_BEGIN,DTMF_END); view only DTMF frames.

exten => 1,1,Set(FRAME_TRACE(black)=DTMF_BEGIN,DTMF_END); view everything except DTMF frames.

Syntax

```
FRAME_TRACE(filter list type)
```

Arguments

- `filter list type` - A filter can be applied to the trace to limit what frames are viewed. This filter can either be a `white` or `black` list of frame types. When no filter type is present, `white` is used. If no arguments are provided at all, all frames will be output. Below are the different types of frames that can be filtered.

- DTMF_BEGIN
- DTMF_END
- VOICE
- VIDEO
- CONTROL
- NULL
- IAX
- TEXT
- IMAGE
- HTML
- CNG
- MODEM

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function GLOBAL

GLOBAL()

Synopsis

Gets or sets the global variable specified.

Description

Set or get the value of a global variable specified in *varname*

Syntax

```
GLOBAL(varname)
```

Arguments

- *varname* - Global variable name

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_HANGUPCAUSE

HANGUPCAUSE()

Synopsis

Gets per-channel hangupcause information from the channel.

Description

Gets technology-specific or translated Asterisk cause code information from the channel for the specified channel that resulted from a dial.

Syntax

```
HANGUPCAUSE(channel,type)
```

Arguments

- `channel` - The name of the channel for which to retrieve cause information.
- `type` - Parameter describing which type of information is requested. Types are:
 - `tech` - Technology-specific cause information
 - `ast` - Translated Asterisk cause code

See Also

- [Asterisk 12 Function_HANGUPCAUSE_KEYS](#)
- [Asterisk 12 Application_HangupCauseClear](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_HANGUPCAUSE_KEYS

HANGUPCAUSE_KEYS()

Synopsis

Gets the list of channels for which hangup causes are available.

Description

Returns a comma-separated list of channel names to be used with the HANGUPCAUSE function.

Syntax

See Also

- [Asterisk 12 Function_HANGUPCAUSE](#)
- [Asterisk 12 Application_HangupCauseClear](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_HINT

HINT()

Synopsis

Get the devices set for a dialplan hint.

Description

The HINT function can be used to retrieve the list of devices that are mapped to a dialplan hint. For example:

NoOp(Hint for Extension 1234 is \${HINT(1234)})

Syntax

```
HINT(extension,options)
```

Arguments

- extension
 - extension
 - context
- options
 - n - Retrieve name on the hint instead of list of devices.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_IAXPEER

IAXPEER()

Synopsis

Gets IAX peer information.

Description

Gets information associated with the specified IAX2 peer.

Syntax

```
IAXPEER(peername,item)
```

Arguments

- `peername`
 - `CURRENTCHANNEL` - If *peername* is specified to this value, return the IP address of the endpoint of the current channel
- `item` - If *peername* is specified, valid items are:
 - `ip` - (default) The IP address.
 - `status` - The peer's status (if `qualify=yes`)
 - `mailbox` - The configured mailbox.
 - `context` - The configured context.
 - `expire` - The epoch time of the next expire.
 - `dynamic` - Is it dynamic? (yes/no).
 - `callerid_name` - The configured Caller ID name.
 - `callerid_num` - The configured Caller ID number.
 - `codecs` - The configured codecs.
 - `codecx` - Preferred codec index number *x* (beginning with 0)

See Also

- [Asterisk 12 Function_SIPPEER](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_IAXVAR

IAXVAR()

Synopsis

Sets or retrieves a remote variable.

Description

Gets or sets a variable that is sent to a remote IAX2 peer during call setup.

Syntax

```
IAXVAR(varname)
```

Arguments

- varname

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_ICONV

ICONV()

Synopsis

Converts charsets of strings.

Description

Converts string from *in-charset* into *out-charset*. For available charsets, use `iconv -l` on your shell command line.



Note

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

Syntax

```
ICONV(in-charset,out-charset,string)
```

Arguments

- *in-charset* - Input charset
- *out-charset* - Output charset
- *string* - String to convert, from *in-charset* to *out-charset*

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_IF

IF()

Synopsis

Check for an expresion.

Description

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

Syntax

```
IF(expression?retvalue)
```

Arguments

- expression
- retvalue
 - true
 - false

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_IFTIME

IFTIME()

Synopsis

Temporal Conditional.

Description

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

Syntax

```
IFTIME(timespec?retvalue)
```

Arguments

- timespec
- retvalue
 - true
 - false

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_IMPORT

IMPORT()

Synopsis

Retrieve the value of a variable from another channel.

Description

Syntax

```
IMPORT(channel,variable)
```

Arguments

- channel
- variable

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_ISNULL

ISNULL()

Synopsis

Check if a value is NULL.

Description

Returns 1 if NULL or 0 otherwise.

Syntax

```
ISNULL(data)
```

Arguments

- data

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_JABBER_RECEIVE_res_jabber

JABBER_RECEIVE() - [res_jabber]

Synopsis

Reads XMPP messages.

Description

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

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

Syntax

```
JABBER_RECEIVE(account,jid,timeout)
```

Arguments

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

See Also

- [Asterisk 12 Function_JABBER_STATUS_res_jabber](#)
- [Asterisk 12 Application_JabberSend_res_jabber](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_JABBER_RECEIVE_res_xmpp

JABBER_RECEIVE() - [res_xmpp]

Synopsis

Reads XMPP messages.

Description

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

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

Syntax

```
JABBER_RECEIVE(account,jid,timeout)
```

Arguments

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

See Also

- [Asterisk 12 Function_JABBER_STATUS_res_xmpp](#)
- [Asterisk 12 Application_JabberSend_res_xmpp](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_JABBER_STATUS_res_jabber

JABBER_STATUS() - [res_jabber]

Synopsis

Retrieves a buddy's status.

Description

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

Status will be 1-7.

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

If not in roster variable will be set to 7.

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

Syntax

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

Arguments

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

See Also

- [Asterisk 12 Function_JABBER_RECEIVE_res_jabber](#)
- [Asterisk 12 Application_JabberSend_res_jabber](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_JABBER_STATUS_res_xmpp

JABBER_STATUS() - [res_xmpp]

Synopsis

Retrieves a buddy's status.

Description

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

Status will be 1-7.

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

If not in roster variable will be set to 7.

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

Syntax

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

Arguments

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

See Also

- [Asterisk 12 Function_JABBER_RECEIVE_res_xmpp](#)
- [Asterisk 12 Application_JabberSend_res_xmpp](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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.

exten => 1,n,Set(JITTERBUFFER(disabled)=);Remove previously applied jitterbuffer



Note

If a channel specifies a jitterbuffer due to channel driver configuration and the JITTERBUFFER function has set a jitterbuffer for that channel, the jitterbuffer set by the JITTERBUFFER function will take priority and the jitterbuffer set by the channel configuration will not be applied.

Syntax

```
JITTERBUFFER(jitterbuffer type)
```

Arguments

- jitterbuffer type - Jitterbuffer type can be fixed, adaptive, or disabled.
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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_LISTFILTER

LISTFILTER()

Synopsis

Remove an item from a list, by name.

Description

Remove *value* from the list contained in the *varname* variable, where the list delimiter is specified by the *delim* parameter. This is very useful for removing a single channel name from a list of channels, for example.

Syntax

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

Arguments

- *varname*
- *delim*
- *value*

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_LOCAL

LOCAL()

Synopsis

Manage variables local to the gosub stack frame.

Description

Read and write a variable local to the gosub stack frame, once we Return() it will be lost (or it will go back to whatever value it had before the Gosub()).

Syntax

```
LOCAL(varname)
```

Arguments

- varname

See Also

- [Asterisk 12 Application_Gosub](#)
- [Asterisk 12 Application_Gosublf](#)
- [Asterisk 12 Application_Return](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_LOCAL_PEEK

LOCAL_PEEK()

Synopsis

Retrieve variables hidden by the local gosub stack frame.

Description

Read a variable *varname* hidden by *n* levels of gosub stack frames. Note that `${LOCAL_PEEK(0,foo)}` is the same as `foo`, since the value of *n* peeks under 0 levels of stack frames; in other words, 0 is the current level. If *n* exceeds the available number of stack frames, then an empty string is returned.

Syntax

```
LOCAL_PEEK(n,varname)
```

Arguments

- *n*
- *varname*

See Also

- [Asterisk 12 Application_Gosub](#)
- [Asterisk 12 Application_Gosublf](#)
- [Asterisk 12 Application_Return](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_LOCK

LOCK()

Synopsis

Attempt to obtain a named mutex.

Description

Attempts to grab a named lock exclusively, and prevents other channels from obtaining the same lock. LOCK will wait for the lock to become available. Returns 1 if the lock was obtained or 0 on error.



Note

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



Note

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
LOCK(lockname)
```

Arguments

- lockname

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 Function_MAILBOX_EXISTS

MAILBOX_EXISTS()

Synopsis

Tell if a mailbox is configured.

Description

**Note**

DEPRECATED. Use VM_INFO(mailbox[@context],exists) instead.

Returns a boolean of whether the corresponding *mailbox* exists. If *context* is not specified, defaults to the `default` context.

Syntax

```
MAILBOX_EXISTS(mailbox@context)
```

Arguments

- `mailbox`
- `context`

See Also

- [Asterisk 12 Function_VM_INFO](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `_MASTER_CHANNEL`

`MASTER_CHANNEL()`

Synopsis

Gets or sets variables on the master channel

Description

Allows access to the channel which created the current channel, if any. If the channel is already a master channel, then accesses local channel variables.

Syntax

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_MATH

MATH()

Synopsis

Performs Mathematical Functions.

Description

Performs mathematical functions based on two parameters and an operator. The returned value type is *type*

Example: Set(i=\${MATH(123%16,int)}) - sets var i=11

Syntax

```
MATH(expression,type)
```

Arguments

- *expression* - Is of the form: *number1opnumber2* where the possible values for *op* are:
+, -, /, *, %, <, >, ^, AND, OR, XOR, <=, >=, == (and behave as their C equivalents)
- *type* - Wanted type of result:
f, float - float(default)
i, int - integer
h, hex - hex
c, char - char

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 12 Application_MeetMe](#)
- [Asterisk 12 Application_MeetMeCount](#)
- [Asterisk 12 Application_MeetMeAdmin](#)
- [Asterisk 12 Application_MeetMeChannelAdmin](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_MESSAGE

MESSAGE()

Synopsis

Create a message or read fields from a message.

Description

This function will read from or write a value to a text message. It is used both to read the data out of an incoming message, as well as modify or create a message that will be sent outbound.

Syntax

```
MESSAGE( argument )
```

Arguments

- `argument` - Field of the message to get or set.
 - `to` - Read-only. The destination of the message. When processing an incoming message, this will be set to the destination listed as the recipient of the message that was received by Asterisk.
 - `from` - Read-only. The source of the message. When processing an incoming message, this will be set to the source of the message.
 - `custom_data` - Write-only. Mark or unmark all message headers for an outgoing message. The following values can be set:
 - `mark_all_outbound` - Mark all headers for an outgoing message.
 - `clear_all_outbound` - Unmark all headers for an outgoing message.
 - `body` - Read/Write. The message body. When processing an incoming message, this includes the body of the message that Asterisk received. When `MessageSend()` is executed, the contents of this field are used as the body of the outgoing message. The body will always be UTF-8.

See Also

- [Asterisk 12 Application_MessageSend](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `MESSAGE_DATA`

`MESSAGE_DATA()`

Synopsis

Read or write custom data attached to a message.

Description

This function will read from or write a value to a text message. It is used both to read the data out of an incoming message, as well as modify a message that will be sent outbound.



Note

If you want to set an outbound message to carry data in the current message, do `Set(MESSAGE_DATA(key)=${MESSAGE_DATA(key)})`.

Syntax

```
MESSAGE_DATA ( argument )
```

Arguments

- `argument` - Field of the message to get or set.

See Also

- [Asterisk 12 Application_MessageSend](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_MINIVMACCOUNT

MINIVMACCOUNT()

Synopsis

Gets MiniVoicemail account information.

Description

Syntax

```
MINIVMACCOUNT(account:item)
```

Arguments

- `account`
- `item` - Valid items are:
 - `path` - Path to account mailbox (if account exists, otherwise temporary mailbox).
 - `hasaccount` - 1 is static Minivm account exists, 0 otherwise.
 - `fullname` - Full name of account owner.
 - `email` - Email address used for account.
 - `etemplate` - Email template for account (default template if none is configured).
 - `ptemplate` - Pager template for account (default template if none is configured).
 - `accountcode` - Account code for the voicemail account.
 - `pincode` - Pin code for voicemail account.
 - `timezone` - Time zone for voicemail account.
 - `language` - Language for voicemail account.
 - `<channel variable name>` - Channel variable value (set in configuration for account).

See Also

- [Asterisk 12 Application_MinivmRecord](#)
- [Asterisk 12 Application_MinivmGreet](#)
- [Asterisk 12 Application_MinivmNotify](#)
- [Asterisk 12 Application_MinivmDelete](#)
- [Asterisk 12 Application_MinivmAccMess](#)
- [Asterisk 12 Application_MinivmMWI](#)
- [Asterisk 12 Function_MINIVMCOUNTER](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_MINIVMCOUNTER

MINIVMCOUNTER()

Synopsis

Reads or sets counters for MiniVoicemail message.

Description

The operation is atomic and the counter is locked while changing the value. The counters are stored as text files in the minivm account directories. It might be better to use realtime functions if you are using a database to operate your Asterisk.

Syntax

```
MINIVMCOUNTER(account:name:operand)
```

Arguments

- **account** - If account is given and it exists, the counter is specific for the account.
If account is a domain and the domain directory exists, counters are specific for a domain.
- **name** - The name of the counter is a string, up to 10 characters.
- **operand** - The counters never goes below zero. Valid operands for changing the value of a counter when assigning a value are:
 - **i** - Increment by value.
 - **d** - Decrement by value.
 - **s** - Set to value.

See Also

- [Asterisk 12 Application_MinivmRecord](#)
- [Asterisk 12 Application_MinivmGreet](#)
- [Asterisk 12 Application_MinivmNotify](#)
- [Asterisk 12 Application_MinivmDelete](#)
- [Asterisk 12 Application_MinivmAccMess](#)
- [Asterisk 12 Application_MinivmMWI](#)
- [Asterisk 12 Function_MINIVMACCOUNT](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_MUTEAUDIO

MUTEAUDIO()

Synopsis

Muting audio streams in the channel

Description

The MUTEAUDIO function can be used to mute inbound (to the PBX) or outbound audio in a call.

Examples:

MUTEAUDIO(in)=on

MUTEAUDIO(in)=off

Syntax

MUTEAUDIO(direction)

Arguments

- `direction` - Must be one of
 - `in` - Inbound stream (to the PBX)
 - `out` - Outbound stream (from the PBX)
 - `all` - Both streams

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_ODBC

ODBC()

Synopsis

Controls ODBC transaction properties.

Description

The ODBC() function allows setting several properties to influence how a connected database processes transactions.

Syntax

```
ODBC(property[,argument])
```

Arguments

- `property`
 - `transaction` - Gets or sets the active transaction ID. If set, and the transaction ID does not exist and a *database name* is specified as an argument, it will be created.
 - `forcecommit` - Controls whether a transaction will be automatically committed when the channel hangs up. Defaults to false. If a *transaction ID* is specified in the optional argument, the property will be applied to that ID, otherwise to the current active ID.
 - `isolation` - Controls the data isolation on uncommitted transactions. May be one of the following: `read_committed`, `read_uncommitted`, `repeatable_read`, or `serializable`. Defaults to the database setting in `res_odbc.conf` or `read_committed` if not specified. If a *transaction ID* is specified as an optional argument, it will be applied to that ID, otherwise the current active ID.
- `argument`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_ODBC_FETCH

ODBC_FETCH()

Synopsis

Fetch a row from a multirow query.

Description

For queries which are marked as mode=multirow, the original query returns a *result-id* from which results may be fetched. This function implements the actual fetch of the results.

This also sets ODBC_FETCH_STATUS.

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

Syntax

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

Arguments

- result-id

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_PASSTHRU

PASSTHRU()

Synopsis

Pass the given argument back as a value.

Description

Literally returns the given *string*. The intent is to permit other dialplan functions which take a variable name as an argument to be able to take a literal string, instead.



Note

The functions which take a variable name need to be passed var and not \${var}. Similarly, use PASSTHRU() and not \${PASSTHRU()}.

Example: \${CHANNEL} contains SIP/321-1

\${CUT(PASSTHRU(\${CUT(CHANNEL,-,1)}),/,2)} will return 321

Syntax

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

Arguments

- string

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_PJSIP_DIAL_CONTACTS

PJSIP_DIAL_CONTACTS()

Synopsis

Return a dial string for dialing all contacts on an AOR.

Description

Returns a properly formatted dial string for dialing all contacts on an AOR.

Syntax

```
PJSIP_DIAL_CONTACTS(endpoint[,aor[,request_user]])
```

Arguments

- `endpoint` - Name of the endpoint
- `aor` - Name of an AOR to use, if not specified the configured AORs on the endpoint are used
- `request_user` - Optional request user to use in the request URI

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_PJSIP_ENDPOINT

PJSIP_ENDPOINT()

Synopsis

Get information about a PJSIP endpoint

Description

Syntax

```
PJSIP_ENDPOINT(name,field)
```

Arguments

- **name** - The name of the endpoint to query.
- **field** - The configuration option for the endpoint to query for. Supported options are those fields on the *endpoint* object in `pjsip.conf`.
 - `100rel` - Allow support for RFC3262 provisional ACK tags
 - `aggregate_mwi` -
 - `allow` - Media Codec(s) to allow
 - `aors` - AoR(s) to be used with the endpoint
 - `auth` - Authentication Object(s) associated with the endpoint
 - `callerid` - CallerID information for the endpoint
 - `callerid_privacy` - Default privacy level
 - `callerid_tag` - Internal `id_tag` for the endpoint
 - `context` - Dialplan context for inbound sessions
 - `direct_media_glare_mitigation` - Mitigation of direct media (re)INVITE glare
 - `direct_media_method` - Direct Media method type
 - `connected_line_method` - Connected line method type
 - `direct_media` - Determines whether media may flow directly between endpoints.
 - `disable_direct_media_on_nat` - Disable direct media session refreshes when NAT obstructs the media session
 - `disallow` - Media Codec(s) to disallow
 - `dtmf_mode` - DTMF mode
 - `media_address` - IP address used in SDP for media handling
 - `force_rport` - Force use of return port
 - `ice_support` - Enable the ICE mechanism to help traverse NAT
 - `identify_by` - Way(s) for Endpoint to be identified
 - `redirect_method` - How redirects received from an endpoint are handled
 - `mailboxes` - Mailbox(es) to be associated with
 - `moh_suggest` - Default Music On Hold class
 - `outbound_auth` - Authentication object used for outbound requests
 - `outbound_proxy` - Proxy through which to send requests, a full SIP URI must be provided
 - `rewrite_contact` - Allow Contact header to be rewritten with the source IP address-port
 - `rtp_ipv6` - Allow use of IPv6 for RTP traffic
 - `rtp_symmetric` - Enforce that RTP must be symmetric
 - `send_diversion` - Send the Diversion header, conveying the diversion information to the called user agent
 - `send_pai` - Send the P-Asserted-Identity header
 - `send_rpid` - Send the Remote-Party-ID header
 - `timers_min_se` - Minimum session timers expiration period
 - `timers` - Session timers for SIP packets
 - `timers_sess_expires` - Maximum session timer expiration period
 - `transport` - Desired transport configuration
 - `trust_id_inbound` - Accept identification information received from this endpoint
 - `trust_id_outbound` - Send private identification details to the endpoint.
 - `type` - Must be of type 'endpoint'.
 - `use_ptime` - Use Endpoint's requested packetisation interval
 - `use_avpf` - Determines whether `res_pjsip` will use and enforce usage of AVPF for this endpoint.

- `media_encryption` - Determines whether `res_pjsip` will use and enforce usage of media encryption for this endpoint.
- `inband_progress` - Determines whether `chan_pjsip` will indicate ringing using inband progress.
- `call_group` - The numeric pickup groups for a channel.
- `pickup_group` - The numeric pickup groups that a channel can pickup.
- `named_call_group` - The named pickup groups for a channel.
- `named_pickup_group` - The named pickup groups that a channel can pickup.
- `device_state_busy_at` - The number of in-use channels which will cause busy to be returned as device state
- `t38_udptl` - Whether T.38 UDPTL support is enabled or not
- `t38_udptl_ec` - T.38 UDPTL error correction method
- `t38_udptl_maxdatagram` - T.38 UDPTL maximum datagram size
- `fax_detect` - Whether CNG tone detection is enabled
- `t38_udptl_nat` - Whether NAT support is enabled on UDPTL sessions
- `t38_udptl_ipv6` - Whether IPv6 is used for UDPTL Sessions
- `tone_zone` - Set which country's indications to use for channels created for this endpoint.
- `language` - Set the default language to use for channels created for this endpoint.
- `one_touch_recording` - Determines whether one-touch recording is allowed for this endpoint.
- `record_on_feature` - The feature to enact when one-touch recording is turned on.
- `record_off_feature` - The feature to enact when one-touch recording is turned off.
- `rtp_engine` - Name of the RTP engine to use for channels created for this endpoint
- `allow_transfer` - Determines whether SIP REFER transfers are allowed for this endpoint
- `sdp_owner` - String placed as the username portion of an SDP origin (o=) line.
- `sdp_session` - String used for the SDP session (s=) line.
- `tos_audio` - DSCP TOS bits for audio streams
- `tos_video` - DSCP TOS bits for video streams
- `cos_audio` - Priority for audio streams
- `cos_video` - Priority for video streams
- `allow_subscribe` - Determines if endpoint is allowed to initiate subscriptions with Asterisk.
- `sub_min_expiry` - The minimum allowed expiry time for subscriptions initiated by the endpoint.
- `from_user` - Username to use in From header for requests to this endpoint.
- `mwi_from_user` - Username to use in From header for unsolicited MWI NOTIFYs to this endpoint.
- `from_domain` - Domain to user in From header for requests to this endpoint.
- `dtls_verify` - Verify that the provided peer certificate is valid
- `dtls_rekey` - Interval at which to renegotiate the TLS session and rekey the SRTP session
- `dtls_cert_file` - Path to certificate file to present to peer
- `dtls_private_key` - Path to private key for certificate file
- `dtls_cipher` - Cipher to use for DTLS negotiation
- `dtls_ca_file` - Path to certificate authority certificate
- `dtls_ca_path` - Path to a directory containing certificate authority certificates
- `dtls_setup` - Whether we are willing to accept connections, connect to the other party, or both.
- `srtp_tag_32` - Determines whether 32 byte tags should be used instead of 80 byte tags.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403680

Asterisk 12 Function_PJSIP_HEADER

PJSIP_HEADER()

Synopsis

Gets, adds, updates or removes the specified SIP header from a PJSIP session.

Description

Examples:

```
;
; Set 'somevar' to the value of the 'From' header.
exten => 1,1,Set(somevar=${PJSIP_HEADER(read,From)})
;
; Set 'via2' to the value of the 2nd 'Via' header.
exten => 1,1,Set(via2=${PJSIP_HEADER(read,Via,2)})
;
; Add an 'X-Myheader' header with the value of 'myvalue'.
exten => 1,1,Set(PJSIP_HEADER(add,X-MyHeader)=myvalue)
;
; Add an 'X-Myheader' header with an empty value.
exten => 1,1,Set(PJSIP_HEADER(add,X-MyHeader)=)
;
; Update the value of the header named 'X-Myheader' to 'newvalue'.
; 'X-Myheader' must already exist or the call will fail.
exten => 1,1,Set(PJSIP_HEADER(update,X-MyHeader)=newvalue)
;
; Remove all headers whose names exactly match 'X-MyHeader'.
exten => 1,1,Set(PJSIP_HEADER(remove,X-MyHeader)=)
;
; Remove all headers that begin with 'X-My'.
exten => 1,1,Set(PJSIP_HEADER(remove,X-My*)=)
;
; Remove all previously added headers.
exten => 1,1,Set(PJSIP_HEADER(remove,*)=)
;
```



Note

The `remove` action can be called by reading **or** writing `PJSIP_HEADER`.

```
;
; Display the number of headers removed
exten => 1,1,Verbose( Removed ${PJSIP_HEADER(remove,X-MyHeader)} headers)
```

```

;

; Set a variable to the number of headers removed

exten => 1,1,Set(count=${PJSIP_HEADER(remove,X-MyHeader)})

;

; Just remove them ignoring any count

exten => 1,1,Set(=${PJSIP_HEADER(remove,X-MyHeader)})

exten => 1,1,Set(PJSIP_HEADER(remove,X-MyHeader)=)

;

```



Note

If you call PJSIP_HEADER in a normal dialplan context you'll be operating on the **caller's (incoming)** channel which may not be what you want. To operate on the **callee's (outgoing)** channel call PJSIP_HEADER in a pre-dial handler.

Example:

```

;

[handler]

exten => addheader,1,Set(PJSIP_HEADER(add,X-MyHeader)=myvalue)

exten => addheader,2,Set(PJSIP_HEADER(add,X-MyHeader2)=myvalue2)

;

[somecontext]

exten => 1,1,Dial(PJSIP/${EXTEN},,b(handler^addheader^1))

;

```

Syntax

```
PJSIP_HEADER(action,name[,number])
```

Arguments

- **action**
 - **read** - Returns instance *number* of header *name*.
 - **add** - Adds a new header *name* to this session.
 - **update** - Updates instance *number* of header *name* to a new value. The header must already exist.
 - **remove** - Removes all instances of previously added headers whose names match *name*. A **{}** may be appended to *name* to **remove all headers *beginning with name**. *name* may be set to a single **{}** to **clear *all** previously added headers. In all cases, the number of headers actually removed is returned.
- **name** - The name of the header.
- **number** - If there's more than 1 header with the same name, this specifies which header to read or update. If not specified, defaults to 1 meaning the first matching header. Not valid for **add** or **remove**.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402154

Asterisk 12 Function_PJSIP_MEDIA_OFFER

PJSIP_MEDIA_OFFER()

Synopsis

Media and codec offerings to be set on an outbound SIP channel prior to dialing.

Description

Returns the codecs offered based upon the media choice

Syntax

```
PJSIP_MEDIA_OFFER(media)
```

Arguments

- `media` - types of media offered

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function `_PRESENCE_STATE`

`PRESENCE_STATE()`

Synopsis

Get or Set a presence state.

Description

The `PRESENCE_STATE` function can be used to retrieve the presence from any presence provider. For example:

```
NoOp(SIP/mypeer has presence ${PRESENCE_STATE(SIP/mypeer,value)})
```

```
NoOp(Conference number 1234 has presence message ${PRESENCE_STATE(MeetMe:1234,message)})
```

The `PRESENCE_STATE` function can also be used to set custom presence state from the dialplan. The `CustomPresence:` prefix must be used. For example:

```
Set(PRESENCE_STATE(CustomPresence:lamp1)=away,temporary,Out to lunch)
```

```
Set(PRESENCE_STATE(CustomPresence:lamp2)=dnd,,Trying to get work done)
```

```
Set(PRESENCE_STATE(CustomPresence:lamp3)=xa,T24gdmFjYXRpb24=,,e)
```

```
Set(BASE64_LAMP3_PRESENCE=${PRESENCE_STATE(CustomPresence:lamp3,subtype,e)})
```

You can subscribe to the status of a custom presence state using a hint in the dialplan:

```
exten => 1234,hint,CustomPresence:lamp1
```

The possible values for both uses of this function are:

not_set | unavailable | available | away | xa | chat | dnd

Syntax

```
PRESENCE_STATE(provider,field[,options])
```

Arguments

- `provider` - The provider of the presence, such as `CustomPresence`
- `field` - Which field of the presence state information is wanted.
 - `value` - The current presence, such as `away`
 - `subtype` - Further information about the current presence
 - `message` - A custom message that may indicate further details about the presence
- `options`
 - `e` - On Write - Use this option when the subtype and message provided are Base64 encoded. On Read - Retrieves message/subtype in Base64 encoded form.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_QUEUE_EXISTS

QUEUE_EXISTS()

Synopsis

Check if a named queue exists on this server

Description

Returns 1 if the specified queue exists, 0 if it does not

Syntax

```
QUEUE_EXISTS (queuename)
```

Arguments

- queuename

See Also

- [Asterisk 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_QUEUE_MEMBER

QUEUE_MEMBER()

Synopsis

Count number of members answering a queue.

Description

Allows access to queue counts [R] and member information [R/W].

queuename is required for all operations *interface* is required for all member operations.

Syntax

```
QUEUE_MEMBER(queuename,option[,interface])
```

Arguments

- *queuename*
- *option*
 - *logged* - Returns the number of logged-in members for the specified queue.
 - *free* - Returns the number of logged-in members for the specified queue that either can take calls or are currently wrapping up after a previous call.
 - *ready* - Returns the number of logged-in members for the specified queue that are immediately available to answer a call.
 - *count* - Returns the total number of members for the specified queue.
 - *penalty* - Gets or sets queue member penalty.
 - *paused* - Gets or sets queue member paused status.
 - *ringinuse* - Gets or sets queue member ringinuse.
- *interface*

See Also

- [Asterisk 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_QUEUE_MEMBER_COUNT

QUEUE_MEMBER_COUNT()

Synopsis

Count number of members answering a queue.

Description

Returns the number of members currently associated with the specified *queuename*.



Warning

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

Syntax

```
QUEUE_MEMBER_COUNT(queuename)
```

Arguments

- `queuename`

See Also

- [Asterisk 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_QUEUE_MEMBER_LIST

QUEUE_MEMBER_LIST()

Synopsis

Returns a list of interfaces on a queue.

Description

Returns a comma-separated list of members associated with the specified *queuename*.

Syntax

```
QUEUE_MEMBER_LIST(queuename)
```

Arguments

- `queuename`

See Also

- [Asterisk 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_QUEUE_MEMBER_PENALTY

QUEUE_MEMBER_PENALTY()

Synopsis

Gets or sets queue members penalty.

Description

Gets or sets queue members penalty.



Warning

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

Syntax

```
QUEUE_MEMBER_PENALTY(queueName,interface)
```

Arguments

- queueName
- interface

See Also

- [Asterisk 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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(queue_name)
```

Arguments

- queue_name
 - QUEUEMAX - Maximum number of calls allowed.
 - QUEUSTRATEGY - 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 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_QUEUE_WAITING_COUNT

QUEUE_WAITING_COUNT()

Synopsis

Count number of calls currently waiting in a queue.

Description

Returns the number of callers currently waiting in the specified *queuename*.

Syntax

```
QUEUE_WAITING_COUNT(queuename)
```

Arguments

- *queuename*

See Also

- [Asterisk 12 Application_Queue](#)
- [Asterisk 12 Application_QueueLog](#)
- [Asterisk 12 Application_AddQueueMember](#)
- [Asterisk 12 Application_RemoveQueueMember](#)
- [Asterisk 12 Application_PauseQueueMember](#)
- [Asterisk 12 Application_UnpauseQueueMember](#)
- [Asterisk 12 Function_QUEUE_VARIABLES](#)
- [Asterisk 12 Function_QUEUE_MEMBER](#)
- [Asterisk 12 Function_QUEUE_MEMBER_COUNT](#)
- [Asterisk 12 Function_QUEUE_EXISTS](#)
- [Asterisk 12 Function_QUEUE_WAITING_COUNT](#)
- [Asterisk 12 Function_QUEUE_MEMBER_LIST](#)
- [Asterisk 12 Function_QUEUE_MEMBER_PENALTY](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_QUOTE

QUOTE()

Synopsis

Quotes a given string, escaping embedded quotes as necessary

Description

Example: `${QUOTE(ab"cd"e)}` will return `"ab\"cd\"e"`

Syntax

```
QUOTE(string)
```

Arguments

- `string`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404099

Asterisk 12 Function_RAND

RAND()

Synopsis

Choose a random number in a range.

Description

Choose a random number between *min* and *max*. *min* defaults to 0, if not specified, while *max* defaults to `RAND_MAX` (2147483647 on many systems).

Example: `Set(junky=${RAND(1,8)})`; Sets junky to a random number between 1 and 8, inclusive.

Syntax

```
RAND(min,max)
```

Arguments

- *min*
- *max*

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_REALTIME

REALTIME()

Synopsis

RealTime Read/Write Functions.

Description

This function will read or write values from/to a RealTime repository. `REALTIME(...)` will read names/values from the repository, and `REALTIME(...)=` will write a new value/field to the repository. On a read, this function returns a delimited text string. The name/value pairs are delimited by *delim1*, and the name and value are delimited between each other with *delim2*. If there is no match, NULL will be returned by the function. On a write, this function will always return NULL.

Syntax

```
REALTIME(family,fieldmatch,matchvalue,delim1|field,delim2)
```

Arguments

- `family`
- `fieldmatch`
- `matchvalue`
- `delim1|field` - Use *delim1* with *delim2* on read and *field* without *delim2* on write
If we are reading and *delim1* is not specified, defaults to ,
- `delim2` - Parameter only used when reading, if not specified defaults to =

See Also

- [Asterisk 12 Function_REALTIME_STORE](#)
- [Asterisk 12 Function_REALTIME_DESTROY](#)
- [Asterisk 12 Function_REALTIME_FIELD](#)
- [Asterisk 12 Function_REALTIME_HASH](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_REALTIME_DESTROY

REALTIME_DESTROY()

Synopsis

RealTime Destroy Function.

Description

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



Note

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be read from the dialplan, and not directly from external protocols. It can, however, be executed as a write operation (`REALTIME_DESTROY(family, fieldmatch)=ignored`)

Syntax

```
REALTIME_DESTROY(family,fieldmatch,matchvalue,delim1,delim2)
```

Arguments

- `family`
- `fieldmatch`
- `matchvalue`
- `delim1`
- `delim2`

See Also

- [Asterisk 12 Function_REALTIME](#)
- [Asterisk 12 Function_REALTIME_STORE](#)
- [Asterisk 12 Function_REALTIME_FIELD](#)
- [Asterisk 12 Function_REALTIME_HASH](#)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 Function_REALTIME_FIELD

REALTIME_FIELD()

Synopsis

RealTime query function.

Description

This function retrieves a single item, *fieldname* from the RT engine, where *fieldmatch* contains the value *matchvalue*. When written to, the REALTIME_FIELD() function performs identically to the REALTIME() function.

Syntax

```
REALTIME_FIELD(family,fieldmatch,matchvalue,fieldname)
```

Arguments

- family
- fieldmatch
- matchvalue
- fieldname

See Also

- [Asterisk 12 Function_REALTIME](#)
- [Asterisk 12 Function_REALTIME_STORE](#)
- [Asterisk 12 Function_REALTIME_DESTROY](#)
- [Asterisk 12 Function_REALTIME_HASH](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_REALTIME_HASH

REALTIME_HASH()

Synopsis

RealTime query function.

Description

This function retrieves a single record from the RT engine, where *fieldmatch* contains the value *matchvalue* and formats the output suitably, such that it can be assigned to the HASH() function. The HASH() function then provides a suitable method for retrieving each field value of the record.

Syntax

```
REALTIME_HASH(family,fieldmatch,matchvalue)
```

Arguments

- family
- fieldmatch
- matchvalue

See Also

- [Asterisk 12 Function_REALTIME](#)
- [Asterisk 12 Function_REALTIME_STORE](#)
- [Asterisk 12 Function_REALTIME_DESTROY](#)
- [Asterisk 12 Function_REALTIME_FIELD](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Function_REALTIME](#)
- [Asterisk 12 Function_REALTIME_DESTROY](#)
- [Asterisk 12 Function_REALTIME_FIELD](#)
- [Asterisk 12 Function_REALTIME_HASH](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_REDIRECTING

REDIRECTING()

Synopsis

Gets or sets Redirecting data on the channel.

Description

Gets or sets Redirecting data on the channel.

The allowable values for the *reason* and *orig-reason* fields are the following:

- unknown - Unknown
- cfb - Call Forwarding Busy
- cfnr - Call Forwarding No Reply
- unavailable - Callee is Unavailable
- time_of_day - Time of Day
- dnd - Do Not Disturb
- deflection - Call Deflection
- follow_me - Follow Me
- out_of_order - Called DTE Out-Of-Order
- away - Callee is Away
- cf_dte - Call Forwarding By The Called DTE
- cfu - Call Forwarding Unconditional

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

- unknown - Unknown
- iso8859-1 - ISO8859-1
- withdrawn - Withdrawn
- iso8859-2 - ISO8859-2
- iso8859-3 - ISO8859-3
- iso8859-4 - ISO8859-4
- iso8859-5 - ISO8859-5
- iso8859-7 - ISO8859-7
- bmp - ISO10646 Bmp String
- utf8 - ISO10646 UTF-8 String

Syntax

```
REDIRECTING(datatype,i)
```

Arguments

- datatype - The allowable datatypes are:
 - orig-all
 - orig-name
 - orig-name-valid
 - orig-name-charset
 - orig-name-pres
 - orig-num
 - orig-num-valid
 - orig-num-plan
 - orig-num-pres
 - orig-subaddr
 - orig-subaddr-valid
 - orig-subaddr-type
 - orig-subaddr-odd
 - orig-tag

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

- `priv-to-name-pres`
- `priv-to-num`
- `priv-to-num-valid`
- `priv-to-num-plan`
- `priv-to-num-pres`
- `priv-to-subaddr`
- `priv-to-subaddr-valid`
- `priv-to-subaddr-type`
- `priv-to-subaddr-odd`
- `priv-to-tag`
- `reason`
- `count`
- `i` - If set, this will prevent the channel from sending out protocol messages because of the value being set

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_REPLACE

REPLACE()

Synopsis

Replace a set of characters in a given string with another character.

Description

Iterates through a string replacing all the *find-chars* with *replace-char*. *replace-char* may be either empty or contain one character. If empty, all *find-chars* will be deleted from the output.



Note

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

Syntax

```
REPLACE(varname,find-chars[,replace-char])
```

Arguments

- varname
- find-chars
- replace-char

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_SET

SET()

Synopsis

SET assigns a value to a channel variable.

Description

Syntax

```
SET(varname=value)
```

Arguments

- varname
- value

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_SHA1

SHA1()

Synopsis

Computes a SHA1 digest.

Description

Generate a SHA1 digest via the SHA1 algorythm.

Example: Set(shash=\${SHA1(junk)})

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

Syntax

```
SHA1(data)
```

Arguments

- data - Input string

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_SHARED

SHARED()

Synopsis

Gets or sets the shared variable specified.

Description

Implements a shared variable area, in which you may share variables between channels.

The variables used in this space are separate from the general namespace of the channel and thus `SHARED(foo)` and `foo` represent two completely different variables, despite sharing the same name.

Finally, realize that there is an inherent race between channels operating at the same time, fiddling with each others' internal variables, which is why this special variable namespace exists; it is to remind you that variables in the SHARED namespace may change at any time, without warning. You should therefore take special care to ensure that when using the SHARED namespace, you retrieve the variable and store it in a regular channel variable before using it in a set of calculations (or you might be surprised by the result).

Syntax

```
SHARED(varname,channel)
```

Arguments

- `varname` - Variable name
- `channel` - If not specified will default to current channel. It is the complete channel name: `SIP/12-abcd1234` or the prefix only `SIP/12`.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_SHELL

SHELL()

Synopsis

Executes a command using the system shell and captures its output.

Description

Collects the output generated by a command executed by the system shell

Example: `Set(foo=${SHELL(echo bar)})`



Note

The command supplied to this function will be executed by the system's shell, typically specified in the SHELL environment variable. There are many different system shells available with somewhat different behaviors, so the output generated by this function may vary between platforms.

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
SHELL(command)
```

Arguments

- `command` - The command that the shell should execute.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 Function_SHIFT

SHIFT()

Synopsis

Removes and returns the first item off of a variable containing delimited text

Description

Example:

```
exten => s,1,Set(array=one,two,three)
```

```
exten => s,n,While($["${SET(var=${SHIFT(array)})}" != ""])
```

```
exten => s,n,NoOp(var is ${var})
```

```
exten => s,n,EndWhile
```

This would iterate over each value in array, left to right, and would result in NoOp(var is one), NoOp(var is two), and NoOp(var is three) being executed.

Syntax

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

Arguments

- varname
- delimiter

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_SIP_HEADER

SIP_HEADER()

Synopsis

Gets the specified SIP header from an incoming INVITE message.

Description

Since there are several headers (such as Via) which can occur multiple times, SIP_HEADER takes an optional second argument to specify which header with that name to retrieve. Headers start at offset 1.

Please observe that contents of the SDP (an attachment to the SIP request) can't be accessed with this function.

Syntax

```
SIP_HEADER (name, number)
```

Arguments

- name
- number - If not specified, defaults to 1.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `_SIPCHANINFO`

`SIPCHANINFO()`

Synopsis

Gets the specified SIP parameter from the current channel.

Description

Syntax

```
SIPCHANINFO(item)
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `SIPPEER`

`SIPPEER()`

Synopsis

Gets SIP peer information.

Description

Syntax

```
SIPPEER(peername,item)
```

Arguments

- `peername`
- `item`
 - `ip` - (default) The IP address.
 - `port` - The port number.
 - `mailbox` - The configured mailbox.
 - `context` - The configured context.
 - `expire` - The epoch time of the next expire.
 - `dynamic` - Is it dynamic? (yes/no).
 - `callerid_name` - The configured Caller ID name.
 - `callerid_num` - The configured Caller ID number.
 - `callgroup` - The configured Callgroup.
 - `pickupgroup` - The configured Pickupgroup.
 - `namedcallgroup` - The configured Named Callgroup.
 - `namedpickupgroup` - The configured Named Pickupgroup.
 - `codecs` - The configured codecs.
 - `status` - Status (if qualify=yes).
 - `regexten` - Extension activated at registration.
 - `limit` - Call limit (call-limit).
 - `busylevel` - Configured call level for signalling busy.
 - `curcalls` - Current amount of calls. Only available if call-limit is set.
 - `language` - Default language for peer.
 - `accountcode` - Account code for this peer.
 - `useragent` - Current user agent header used by peer.
 - `maxforwards` - The value used for SIP loop prevention in outbound requests
 - `chanvarname` - A channel variable configured with setvar for this peer.
 - `codecx` - Preferred codec index number *x* (beginning with zero).

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_SMDI_MSG

SMDI_MSG()

Synopsis

Retrieve details about an SMDI message.

Description

This function is used to access details of an SMDI message that was pulled from the incoming SMDI message queue using the SMDI_MSG_RETRIEVE() function.

Syntax

```
SMDI_MSG(message_id,component)
```

Arguments

- `message_id`
- `component` - Valid message components are:
 - `number` - The message desk number
 - `terminal` - The message desk terminal
 - `station` - The forwarding station
 - `callerid` - The callerID of the calling party that was forwarded
 - `type` - The call type. The value here is the exact character that came in on the SMDI link. Typically, example values are:
Options:
 - `D` - Direct Calls
 - `A` - Forward All Calls
 - `B` - Forward Busy Calls
 - `N` - Forward No Answer Calls

See Also

- [Asterisk 12 Function_SMDI_MSG_RETRIEVE](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 12 Function_SMDI_MSG](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_SORT

SORT()

Synopsis

Sorts a list of key/vals into a list of keys, based upon the vals.

Description

Takes a comma-separated list of keys and values, each separated by a colon, and returns a comma-separated list of the keys, sorted by their values. Values will be evaluated as floating-point numbers.

Syntax

```
SORT(keyval,keyvaln[,...])
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function `SPEECH_ENGINE`

`SPEECH_ENGINE()`

Synopsis

Get or change a speech engine specific attribute.

Description

Changes a speech engine specific attribute.

Syntax

```
SPEECH_ENGINE ( name )
```

Arguments

- name

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `SPEECH_GRAMMAR`

`SPEECH_GRAMMAR()`

Synopsis

Gets the matched grammar of a result if available.

Description

Gets the matched grammar of a result if available.

Syntax

```
SPEECH_GRAMMAR(nbest_number/result_number)
```

Arguments

- `nbest_number`
- `result_number`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `SPEECH_RESULTS_TYPE`

`SPEECH_RESULTS_TYPE()`

Synopsis

Sets the type of results that will be returned.

Description

Sets the type of results that will be returned. Valid options are normal or nbest.

Syntax

```
SPEECH_RESULTS_TYPE( )
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function `SPEECH_SCORE`

`SPEECH_SCORE()`

Synopsis

Gets the confidence score of a result.

Description

Gets the confidence score of a result.

Syntax

```
SPEECH_SCORE(nbest_number/result_number)
```

Arguments

- `nbest_number`
- `result_number`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function `_SPRINTF`

`SPRINTF()`

Synopsis

Format a variable according to a format string.

Description

Parses the format string specified and returns a string matching that format. Supports most options found in **`sprintf(3)`**. Returns a shortened string if a format specifier is not recognized.

Syntax

```
SPRINTF(format, arg1, arg2[, ...], argN)
```

Arguments

- `format`
- `arg1`
- `arg2`
- `argN`

See Also

- `sprintf(3)`

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_SQL_ESC

SQL_ESC()

Synopsis

Escapes single ticks for use in SQL statements.

Description

Used in SQL templates to escape data which may contain single ticks ' which are otherwise used to delimit data.

Example: SELECT foo FROM bar WHERE baz='\${SQL_ESC(\${ARG1}})'

Syntax

```
SQL_ESC(string)
```

Arguments

- string

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function `_STACK_PEEK`

`STACK_PEEK()`

Synopsis

View info about the location which called Gosub

Description

Read the calling {{c}}ontext, {{e}}xtension, {{p}}riority, or {{l}}abel, as specified by *which*, by going up *n* frames in the Gosub stack. If *suppress* is true, then if the number of available stack frames is exceeded, then no error message will be printed.

Syntax

```
STACK_PEEK(n,which[,suppress])
```

Arguments

- *n*
- *which*
- *suppress*

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_STAT

STAT()

Synopsis

Does a check on the specified file.

Description



Note

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 Function_STRFTIME

STRFTIME()

Synopsis

Returns the current date/time in the specified format.

Description

STRFTIME supports all of the same formats as the underlying C function **strftime(3)**. It also supports the following format: %[n]q - fractions of a second, with leading zeros.

Example: %3q will give milliseconds and %1q will give tenths of a second. The default is set at milliseconds (n=3). The common case is to use it in combination with %S, as in %S.%3q.

Syntax

```
STRFTIME(epoch,timezone,format)
```

Arguments

- epoch
- timezone
- format

See Also

- `strftime(3)`

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_STRPTIME

STRPTIME()

Synopsis

Returns the epoch of the arbitrary date/time string structured as described by the format.

Description

This is useful for converting a date into `EPOCH` time, possibly to pass to an application like `SayUnixTime` or to calculate the difference between the two date strings

Example: `${STRPTIME(2006-03-01 07:30:35,America/Chicago,%Y-%m-%d %H:%M:%S)}` returns 1141219835

Syntax

```
STRPTIME(datetime,timezone,format)
```

Arguments

- `datetime`
- `timezone`
- `format`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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.



Note

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

Syntax

```
STRREPLACE(varname,find-string[,replace-string[,max-replacements]])
```

Arguments

- varname
- find-string
- replace-string
- max-replacements

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_SYSINFO

SYSINFO()

Synopsis

Returns system information specified by parameter.

Description

Returns information from a given parameter.

Syntax

```
SYSINFO(parameter)
```

Arguments

- parameter
 - loadavg - System load average from past minute.
 - numcalls - Number of active calls currently in progress.
 - uptime - System uptime in hours.

**Note**

This parameter is dependant upon operating system.

- totalram - Total usable main memory size in KiB.

**Note**

This parameter is dependant upon operating system.

- freeram - Available memory size in KiB.

**Note**

This parameter is dependant upon operating system.

- bufferram - Memory used by buffers in KiB.

**Note**

This parameter is dependant upon operating system.

- totalswap - Total swap space still available in KiB.

**Note**

This parameter is dependant upon operating system.

- freeswap - Free swap space still available in KiB.

**Note**

This parameter is dependant upon operating system.

- numprocs - Number of current processes.

**Note**

This parameter is dependant upon operating system.

See Also

Import Version

Asterisk 12 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 12 Application_GotoIfTime](#)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_TIMEOUT

TIMEOUT()

Synopsis

Gets or sets timeouts on the channel. Timeout values are in seconds.

Description

The timeouts that can be manipulated are:

absolute: The absolute maximum amount of time permitted for a call. Setting of 0 disables the timeout.

digit: The maximum amount of time permitted between digits when the user is typing in an extension. When this timeout expires, after the user has started to type in an extension, the extension will be considered complete, and will be interpreted. Note that if an extension typed in is valid, it will not have to timeout to be tested, so typically at the expiry of this timeout, the extension will be considered invalid (and thus control would be passed to the *i* extension, or if it doesn't exist the call would be terminated). The default timeout is 5 seconds.

response: The maximum amount of time permitted after falling through a series of priorities for a channel in which the user may begin typing an extension. If the user does not type an extension in this amount of time, control will pass to the *t* extension if it exists, and if not the call would be terminated. The default timeout is 10 seconds.

Syntax

```
TIMEOUT(timeouttype)
```

Arguments

- `timeouttype` - The timeout that will be manipulated. The possible timeout types are: `absolute`, `digit` or `response`

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_TOLOWER

TOLOWER()

Synopsis

Convert string to all lowercase letters.

Description

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

Syntax

```
TOLOWER(string)
```

Arguments

- string

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_TOUPPER

TOUPPER()

Synopsis

Convert string to all uppercase letters.

Description

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

Syntax

```
TOUPPER(string)
```

Arguments

- string

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_TRYLOCK

TRYLOCK()

Synopsis

Attempt to obtain a named mutex.

Description

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



Note

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
TRYLOCK(lockname)
```

Arguments

- `lockname`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 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 Unknown

Asterisk 12 Function_UNLOCK

UNLOCK()

Synopsis

Unlocks a named mutex.

Description

Unlocks a previously locked mutex. Returns 1 if the channel had a lock or 0 otherwise.



Note

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



Note

If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
UNLOCK(lockname)
```

Arguments

- `lockname`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403959

Asterisk 12 Function_UNSHIFT

UNSHIFT()

Synopsis

Inserts one or more values to the beginning of a variable containing delimited text

Description

Example: Set(UNSHIFT(array)=one,two,three) would insert one, two, and three before the values stored in the variable "array".

Syntax

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

Arguments

- varname
- delimiter

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 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 Unknown

Asterisk 12 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 Unknown

Asterisk 12 Function_VALID_EXTEN

VALID_EXTEN()

Synopsis

Determine whether an extension exists or not.

Description

Returns a true value if the indicated *context*, *extension*, and *priority* exist.



Warning

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

Syntax

```
VALID_EXTEN(context,extension,priority)
```

Arguments

- `context` - Defaults to the current context
- `extension`
- `priority` - Priority defaults to 1.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_VERSION

VERSION()

Synopsis

Return the Version info for this Asterisk.

Description

If there are no arguments, return the version of Asterisk in this format: SVN-branch-1.4-r44830M

Example: Set(junky=\${VERSION()});

Sets junky to the string SVN-branch-1.6-r74830M, or possibly, SVN-trunk-r45126M.

Syntax

```
VERSION(info)
```

Arguments

- `info` - The possible values are:
 - `ASTERISK_VERSION_NUM` - A string of digits is returned, e.g. 10602 for 1.6.2 or 100300 for 10.3.0, or 999999 when using an SVN build.
 - `BUILD_USER` - The string representing the user's name whose account was used to configure Asterisk, is returned.
 - `BUILD_HOSTNAME` - The string representing the name of the host on which Asterisk was configured, is returned.
 - `BUILD_MACHINE` - The string representing the type of machine on which Asterisk was configured, is returned.
 - `BUILD_OS` - The string representing the OS of the machine on which Asterisk was configured, is returned.
 - `BUILD_DATE` - The string representing the date on which Asterisk was configured, is returned.
 - `BUILD_KERNEL` - The string representing the kernel version of the machine on which Asterisk was configured, is returned.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_VM_INFO

VM_INFO()

Synopsis

Returns the selected attribute from a mailbox.

Description

Returns the selected attribute from the specified *mailbox*. If *context* is not specified, defaults to the `default` context. Where the *folder* can be specified, common folders include `INBOX`, `Old`, `Work`, `Family` and `Friends`.

Syntax

```
VM_INFO(mailbox,attribute[,folder])
```

Arguments

- `mailbox`
 - `mailbox`
 - `context`
- `attribute`
 - `count` - Count of messages in specified *folder*. If *folder* is not specified, defaults to `INBOX`.
 - `email` - E-mail address associated with the mailbox.
 - `exists` - Returns a boolean of whether the corresponding *mailbox* exists.
 - `fullname` - Full name associated with the mailbox.
 - `language` - Mailbox language if overridden, otherwise the language of the channel.
 - `locale` - Mailbox locale if overridden, otherwise global locale.
 - `pager` - Pager e-mail address associated with the mailbox.
 - `password` - Mailbox access password.
 - `tz` - Mailbox timezone if overridden, otherwise global timezone
- `folder` - If not specified, `INBOX` is assumed.

See Also

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Function_VMCOUNT

VMCOUNT()

Synopsis

Count the voicemails in a specified mailbox.

Description

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

Example: exten => s,1,Set(foo=\${VMCOUNT(125@default)})

Syntax

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

Arguments

- vmbox
- folder - If not specified, defaults to INBOX

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404375

Asterisk 12 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 Unknown

Asterisk 12 Module Configuration

Asterisk 12 Configuration_app_agent_pool

Agent pool applications

This configuration documentation is for functionality provided by `app_agent_pool`.

Overview

**Note**

Option changes take effect on agent login or after an agent disconnects from a call.

agents.conf

global

Unused, but reserved.

agent-id

Configure an agent for the pool.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>ackcall</code>	Boolean	no	false	Enable to require the agent to acknowledge a call.
<code>acceptdtmf</code>	String	#	false	DTMF key sequence the agent uses to acknowledge a call.
<code>autologoff</code>	Unsigned Integer	0	false	Time the agent has to acknowledge a call before being logged off.
<code>wrapuptime</code>	Unsigned Integer	0	false	Minimum time the agent has between calls.
<code>musiconhold</code>	String	default	false	Music on hold class the agent listens to between calls.
<code>recordagentcalls</code>	Boolean	no	false	Enable to automatically record calls the agent takes.
<code>custom_beep</code>	String	beep	false	Sound file played to alert the agent when a call is present.
<code>fullname</code>	String		false	A friendly name for the agent used in log messages.

Configuration Option Descriptions

ackcall

Enable to require the agent to give a DTMF acknowledgement when the agent receives a call.

**Note**

The option is overridden by `AGENTACKCALL` on agent login.

**Note**

Option changes take effect on agent login or after an agent disconnects from a call.

acceptdtmf**Note**

The option is overridden by `AGENTACCEPTDTMF` on agent login.

**Note**

The option is ignored unless the `ackcall` option is enabled.

**Note**

Option changes take effect on agent login or after an agent disconnects from a call.

autologoff

Set how many seconds a call for the agent has to wait for the agent to acknowledge the call before the agent is automatically logged off. If set to zero then the call will wait forever for the agent to acknowledge.

**Note**

The option is overridden by `AGENTAUTOLOGOFF` on agent login.

**Note**

The option is ignored unless the `ackcall` option is enabled.

**Note**

Option changes take effect on agent login or after an agent disconnects from a call.

wrapuptime

Set the minimum amount of time in milliseconds after disconnecting a call before the agent can receive a new call.

**Note**

The option is overridden by `AGENTWRAPUPTIME` on agent login.

**Note**

Option changes take effect on agent login or after an agent disconnects from a call.

musiconhold**Note**

Option changes take effect on agent login or after an agent disconnects from a call.

recordagentcalls

Enable recording calls the agent takes automatically by invoking the `automixmon` DTMF feature when the agent connects to a caller. See `features.conf.sample` for information about the `automixmon` feature.

**Note**

Option changes take effect on agent login or after an agent disconnects from a call.

custom_beep**Note**

Option changes take effect on agent login or after an agent disconnects from a call.

fullname**Note**

Option changes take effect on agent login or after an agent disconnects from a call.

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Configuration_app_confbridge

Conference Bridge Application

This configuration documentation is for functionality provided by `app_confbridge`.

confbridge.conf

global

Unused, but reserved.

user_profile

A named profile to apply to specific callers.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>type</code>	None		<code>false</code>	Define this configuration category as a user profile.
<code>admin</code>	Boolean	<code>no</code>	<code>false</code>	Sets if the user is an admin or not
<code>marked</code>	Boolean	<code>no</code>	<code>false</code>	Sets if this is a marked user or not
<code>startmuted</code>	Boolean	<code>no</code>	<code>false</code>	Sets if all users should start out muted
<code>music_on_hold_when_empty</code>	Boolean	<code>no</code>	<code>false</code>	Play MOH when user is alone or waiting on a marked user
<code>quiet</code>	Boolean	<code>no</code>	<code>false</code>	Silence enter/leave prompts and user intros for this user
<code>announce_user_count</code>	Boolean	<code>no</code>	<code>false</code>	Sets if the number of users should be announced to the user
<code>announce_user_count_all</code>	Custom	<code>no</code>	<code>false</code>	Announce user count to all the other users when this user joins
<code>announce_only_user</code>	Boolean	<code>yes</code>	<code>false</code>	Announce to a user when they join an empty conference
<code>wait_marked</code>	Boolean	<code>no</code>	<code>false</code>	Sets if the user must wait for a marked user to enter before joining a conference
<code>end_marked</code>	Boolean	<code>no</code>	<code>false</code>	Kick the user from the conference when the last marked user leaves
<code>talk_detection_events</code>	Boolean	<code>no</code>	<code>false</code>	Set whether or not notifications of when a user begins and ends talking should be sent out as events over AMI

dtmf_passthrough	Boolean	no	false	Sets whether or not DTMF should pass through the conference
announce_join_leave	Boolean	no	false	Prompt user for their name when joining a conference and play it to the conference when they enter
pin	String		false	Sets a PIN the user must enter before joining the conference
music_on_hold_class	String		false	The MOH class to use for this user
announcement	String		false	Sound file to play to the user when they join a conference
denoise	Boolean	no	false	Apply a denoise filter to the audio before mixing
dsp_drop_silence	Boolean	no	false	Drop what Asterisk detects as silence from audio sent to the bridge
dsp_silence_threshold	Unsigned Integer	2500	false	The number of milliseconds of detected silence necessary to trigger silence detection
dsp_talking_threshold	Unsigned Integer	160	false	The number of milliseconds of detected non-silence necessary to trigger talk detection
jitterbuffer	Boolean	no	false	Place a jitter buffer on the user's audio stream before audio mixing is performed
template	Custom		false	When using the CONFBRIDGE dialplan function, use a user profile as a template for creating a new temporary profile

Configuration Option Descriptions

type

The type parameter determines how a context in the configuration file is interpreted.

- `user` - Configure the context as a *user_profile*
- `bridge` - Configure the context as a *bridge_profile*
- `menu` - Configure the context as a *menu*

announce_user_count_all

Sets if the number of users should be announced to all the other users in the conference when this user joins. This option can be either set to 'yes' or a number. When set to a number, the announcement will only occur once the user count is above the specified number.

denoise

Sets whether or not a denoise filter should be applied to the audio before mixing or not. Off by default. Requires `codec_speex` to be built and installed. Do not confuse this option with `drop_silence`. Denoise is useful if there is a lot of background noise for a user as it attempts to remove the noise while preserving the speech. This option does NOT remove silence from being mixed into the conference and does come at the cost of a slight performance hit.

dsp_drop_silence

This option drops what Asterisk detects as silence from entering into the bridge. Enabling this option will drastically improve performance and help remove the buildup of background noise from the conference. Highly recommended for large conferences due to its performance enhancements.

dsp_silence_threshold

The time in milliseconds of sound falling within the what the dsp has established as baseline silence before a user is considered be silent. This value affects several operations and should not be changed unless the impact on call quality is fully understood.

What this value affects internally:

1. When talk detection AMI events are enabled, this value determines when the user has stopped talking after a period of talking. If this value is set too low AMI events indicating the user has stopped talking may get falsely sent out when the user briefly pauses during mid sentence.
2. The `drop_silence` option depends on this value to determine when the user's audio should begin to be dropped from the conference bridge after the user stops talking. If this value is set too low the user's audio stream may sound choppy to the other participants. This is caused by the user transitioning constantly from silence to talking during mid sentence.

The best way to approach this option is to set it slightly above the maximum amount of ms of silence a user may generate during natural speech.

By default this value is 2500ms. Valid values are 1 through 2^{31} .

dsp_talking_threshold

The time in milliseconds of sound above what the dsp has established as base line silence for a user before a user is considered to be talking. This value affects several operations and should not be changed unless the impact on call quality is fully understood.

What this value affects internally:

1. Audio is only mixed out of a user's incoming audio stream if talking is detected. If this value is set too loose the user will hear themselves briefly each time they begin talking until the dsp has time to establish that they are in fact talking.
2. When talk detection AMI events are enabled, this value determines when talking has begun which results in an AMI event to fire. If this value is set too tight AMI events may be falsely triggered by variants in room noise.
3. The `drop_silence` option depends on this value to determine when the user's audio should be mixed into the bridge after periods of silence. If this value is too loose the beginning of a user's speech will get cut off as they transition from silence to talking.

By default this value is 160 ms. Valid values are 1 through 2^{31}

jitterbuffer

Enabling this option places a jitterbuffer on the user's audio stream before audio mixing is performed. This is highly recommended but will add a slight delay to the audio. This option is using the `JITTERBUFFER` dialplan function's default adaptive jitterbuffer. For a more fine tuned jitterbuffer, disable this option and use the `JITTERBUFFER` dialplan function on the user before entering the ConfBridge application.

bridge_profile

A named profile to apply to specific bridges.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>type</code>	None		<code>false</code>	Define this configuration category as a bridge profile
<code>jitterbuffer</code>	Boolean	<code>no</code>	<code>false</code>	Place a jitter buffer on the conference's audio stream

<code>internal_sample_rate</code>	Unsigned Integer	0	false	Set the internal native sample rate for mixing the conference
<code>language</code>	String	en	false	The language used for announcements to the conference.
<code>mixing_interval</code>	Custom	20	false	Sets the internal mixing interval in milliseconds for the bridge
<code>record_conference</code>	Boolean	no	false	Record the conference starting with the first active user's entrance and ending with the last active user's exit
<code>record_file</code>	String	confbridge-name of conference bridge-start time.wav	false	The filename of the conference recording
<code>record_file_append</code>	Boolean	yes	false	Append record file when starting/stopping on same conference recording
<code>video_mode</code>	Custom		false	Sets how confbridge handles video distribution to the conference participants
<code>max_members</code>	Unsigned Integer	0	false	Limit the maximum number of participants for a single conference
<code>sound_</code>	Custom		true	Override the various conference bridge sound files
<code>template</code>	Custom		false	When using the CONFBRIDGE dialplan function, use a bridge profile as a template for creating a new temporary profile

Configuration Option Descriptions

type

The type parameter determines how a context in the configuration file is interpreted.

- `user` - Configure the context as a *user_profile*
- `bridge` - Configure the context as a *bridge_profile*
- `menu` - Configure the context as a *menu*

internal_sample_rate

Sets the internal native sample rate the conference is mixed at. This is set to automatically adjust the sample rate to the best quality by default. Other values can be anything from 8000-192000. If a sample rate is set that Asterisk does not support, the closest sample rate Asterisk does support to the one requested will be used.

language

By default, announcements to a conference use English. Which means the prompts played to all users within the conference will be English. By changing the language of a bridge, this will change the language of the prompts played to all users.

mixing_interval

Sets the internal mixing interval in milliseconds for the bridge. This number reflects how tight or loose the mixing will be for the conference. In order to improve performance a larger mixing interval such as 40ms may be chosen. Using a larger mixing interval comes at the cost of introducing larger amounts of delay into the bridge. Valid values here are 10, 20, 40, or 80.

record_conference

Records the conference call starting when the first user enters the room, and ending when the last user exits the room. The default recorded filename is 'confbridge-\${name of conference bridge}-\${start time}.wav' and the default format is 8khz slinear. This file will be located in the configured monitoring directory in `asterisk.conf`.

record_file

When `record_conference` is set to yes, the specific name of the record file can be set using this option. Note that since multiple conferences may use the same bridge profile, this may cause issues depending on the configuration. It is recommended to only use this option dynamically with the `CONFBRIDGE()` dialplan function. This allows the record name to be specified and a unique name to be chosen. By default, the `record_file` is stored in Asterisk's `spool/monitor` directory with a unique filename starting with the 'confbridge' prefix.

record_file_append

When `record_file_append` is set to yes, stopping and starting recording on a conference adds the new portion to end of current `record_file`. When this is set to no, a new `record_file` is generated every time you start then stop recording on a conference.

video_mode

Sets how confbridge handles video distribution to the conference participants. Note that participants wanting to view and be the source of a video feed **MUST** be sharing the same video codec. Also, using video in conjunction with the jitterbuffer currently results in the audio being slightly out of sync with the video. This is a result of the jitterbuffer only working on the audio stream. It is recommended to disable the jitterbuffer when video is used.

- `none` - No video sources are set by default in the conference. It is still possible for a user to be set as a video source via AMI or DTMF action at any time.
- `follow_talker` - The video feed will follow whoever is talking and providing video.
- `last_marked` - The last marked user to join the conference with video capabilities will be the single source of video distributed to all participants. If multiple marked users are capable of video, the last one to join is always the source, when that user leaves it goes to the one who joined before them.
- `first_marked` - The first marked user to join the conference with video capabilities is the single source of video distribution among all participants. If that user leaves, the marked user to join after them becomes the source.

max_members

This option limits the number of participants for a single conference to a specific number. By default conferences have no participant limit. After the limit is reached, the conference will be locked until someone leaves. Note however that an Admin user will always be allowed to join the conference regardless if this limit is reached or not.

sound_

All sounds in the conference are customizable using the bridge profile options below. Simply state the option followed by the filename or full path of the filename after the option. Example: `sound_had_joined=conf-hasjoin` This will play the `conf-hasjoin` sound file found in the sounds directory when announcing someone's name is joining the conference.

- `sound_join` - The sound played to everyone when someone enters the conference.
- `sound_leave` - The sound played to everyone when someone leaves the conference.
- `sound_has_joined` - The sound played before announcing someone's name has joined the conference. This is used for user intros. Example "____ has joined the conference"
- `sound_has_left` - The sound played when announcing someone's name has left the conference. This is used for user intros. Example "____ has left the conference"
- `sound_kicked` - The sound played to a user who has been kicked from the conference.
- `sound_muted` - The sound played when the mute option is toggled on.

- `sound_unmuted` - The sound played when the mute option is toggled off.
- `sound_only_person` - The sound played when the user is the only person in the conference.
- `sound_only_one` - The sound played to a user when there is only one other person in the conference.
- `sound_there_are` - The sound played when announcing how many users there are in a conference.
- `sound_other_in_party` - This file is used in conjunction with `sound_there_are` when announcing how many users there are in the conference. The sounds are strung together like this. `"sound_there_are" ${number of participants}`
`"sound_other_in_party"`
- `sound_place_into_conference` - The sound played when someone is placed into the conference after waiting for a marked user.
- `sound_wait_for_leader` - The sound played when a user is placed into a conference that can not start until a marked user enters.
- `sound_leader_has_left` - The sound played when the last marked user leaves the conference.
- `sound_get_pin` - The sound played when prompting for a conference pin number.
- `sound_invalid_pin` - The sound played when an invalid pin is entered too many times.
- `sound_locked` - The sound played to a user trying to join a locked conference.
- `sound_locked_now` - The sound played to an admin after toggling the conference to locked mode.
- `sound_unlocked_now` - The sound played to an admin after toggling the conference to unlocked mode.
- `sound_error_menu` - The sound played when an invalid menu option is entered.

menu

A conference user menu

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>type</code>	None		<code>false</code>	Define this configuration category as a menu
<code>0-9A-D*#</code>	Custom		<code>true</code>	DTMF sequences to assign various confbridge actions to

Configuration Option Descriptions

type

The type parameter determines how a context in the configuration file is interpreted.

- `user` - Configure the context as a *user_profile*
- `bridge` - Configure the context as a *bridge_profile*
- `menu` - Configure the context as a *menu*

0-9A-D*#

The ConfBridge application also has the ability to apply custom DTMF menus to each channel using the application. Like the User and Bridge profiles a menu is passed in to ConfBridge as an argument in the dialplan.

Below is a list of menu actions that can be assigned to a DTMF sequence.



Note

To have the first DTMF digit in a sequence be the '#' character, you need to escape it. If it is not escaped then normal config file processing will think it is a directive like `#include`. For example: The mute setting is toggled when #1 is pressed.

```
#1=toggle_mute
```



Note

A single DTMF sequence can have multiple actions associated with it. This is accomplished by stringing the actions together and using a , as the delimiter. Example: Both listening and talking volume is reset when 5 is pressed. `5=reset_talking_volume, reset_listening_volume`

- `playback(filename&filename2&...)` - `playback` will play back an audio file to a channel and then immediately return to the conference. This file can not be interrupted by DTMF. Multiple files can be chained together using the `&` character.
- `playback_and_continue(filename&filename2&...)` - `playback_and_continue` will play back a prompt while continuing to collect the dtmf sequence. This is useful when using a menu prompt that describes all the menu options. Note however that any DTMF during this action will terminate the prompts playback. Prompt files can be chained together using the `&` character as a delimiter.
- `toggle_mute` - Toggle turning on and off mute. Mute will make the user silent to everyone else, but the user will still be able to listen in.
- `no_op` - This action does nothing (No Operation). Its only real purpose exists for being able to reserve a sequence in the config as a menu exit sequence.
- `decrease_listening_volume` - Decreases the channel's listening volume.
- `increase_listening_volume` - Increases the channel's listening volume.
- `reset_listening_volume` - Reset channel's listening volume to default level.
- `decrease_talking_volume` - Decreases the channel's talking volume.
- `increase_talking_volume` - Increases the channel's talking volume.
- `reset_talking_volume` - Reset channel's talking volume to default level.
- `dialplan_exec(context,exten,priority)` - The `dialplan_exec` action allows a user to escape from the conference and execute commands in the dialplan. Once the dialplan exits the user will be put back into the conference. The possibilities are endless!
- `leave_conference` - This action allows a user to exit the conference and continue execution in the dialplan.
- `admin_kick_last` - This action allows an Admin to kick the last participant from the conference. This action will only work for admins which allows a single menu to be used for both users and admins.
- `admin_toggle_conference_lock` - This action allows an Admin to toggle locking and unlocking the conference. Non admins can not use this action even if it is in their menu.
- `set_as_single_video_src` - This action allows any user to set themselves as the single video source distributed to all participants. This will make the video feed stick to them regardless of what the `video_mode` is set to.
- `release_as_single_video_src` - This action allows a user to release themselves as the video source. If `video_mode` is not set to `none` this action will result in the conference returning to whatever video mode the bridge profile is using. Note that this action will have no effect if the user is not currently the video source. Also, the user is not guaranteed by using this action that they will not become the video source again. The bridge will return to whatever operation the `video_mode` option is set to upon release of the video src.
- `admin_toggle_mute_participants` - This action allows an administrator to toggle the mute state for all non-admins within a conference. All admin users are unaffected by this option. Note that all users, regardless of their admin status, are notified that the conference is muted.
- `participant_count` - This action plays back the number of participants currently in a conference

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404375

Asterisk 12 Configuration_app_skel

This configuration documentation is for functionality provided by app_skel.

app_skel.conf

globals

Options that apply globally to app_skel

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
games				The number of games a single execution of SkelGuessNumber will play
cheat				Should the computer cheat?

Configuration Option Descriptions

cheat

If enabled, the computer will ignore winning guesses.

sounds

Prompts for SkelGuessNumber to play

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
prompt		please-enter-yournumberqueue-less-than		A prompt directing the user to enter a number less than the max number
wrong_guess		vm-pls-try-again		The sound file to play when a wrong guess is made
right_guess		auth-thankyou		The sound file to play when a correct guess is made
too_low				The sound file to play when a guess is too low
too_high				The sound file to play when a guess is too high
lose		vm-goodbye		The sound file to play when a player loses

level

Defined levels for the SkelGuessNumber game

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
-------------	------	---------------	--------------------	-------------

max_number				The maximum in the range of numbers to guess (1 is the implied minimum)
max_guesses				The maximum number of guesses before a game is considered lost

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404375

Asterisk 12 Configuration_ari

HTTP binding for the Stasis API

This configuration documentation is for functionality provided by `ari`.

ari.conf

general

General configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>enabled</code>				Enable/disable the ARI module
<code>pretty</code>				Responses from ARI are formatted to be human readable
<code>auth_realm</code>				Realm to use for authentication. Defaults to Asterisk REST Interface.
<code>allowed_origins</code>				Comma separated list of allowed origins, for Cross-Origin Resource Sharing. May be set to * to allow all origins.

user

Per-user configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>type</code>				Define this configuration section as a user.
<code>read_only</code>				When set to yes, user is only authorized for read-only requests
<code>password</code>				Crypted or plaintext password (see <code>password_format</code>)
<code>password_format</code>				<code>password_format</code> may be set to plain (the default) or crypt. When set to crypt, <code>crypt(3)</code> is used to validate the password. A crypted password can be generated using <code>mkpasswd -m sha-512</code> . When set to plain, the password is in plaintext

Configuration Option Descriptions

type

- `user` - Configure this section as a *user*

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403304

Asterisk 12 Configuration_cdr

Call Detail Record configuration

This configuration documentation is for functionality provided by `cdr`.

Overview

CDR is Call Detail Record, which provides logging services via a variety of pluggable backend modules. Detailed call information can be recorded to databases, files, etc. Useful for billing, fraud prevention, compliance with Sarbanes-Oxley aka The Enron Act, QOS evaluations, and more.

`cdr.conf`

general

Global settings applied to the CDR engine.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>debug</code>	Boolean		<code>false</code>	Enable/disable verbose CDR debugging.
<code>enable</code>	Boolean	1	<code>false</code>	Enable/disable CDR logging.
<code>unanswered</code>	Boolean	0	<code>false</code>	Log calls that are never answered.
<code>congestion</code>	Boolean		<code>false</code>	Log congested calls.
<code>endbeforehexten</code>	Boolean	0	<code>false</code>	Don't produce CDRs while executing hangup logic
<code>initiatedseconds</code>	Boolean	0	<code>false</code>	Count microseconds for billsec purposes
<code>batch</code>	Boolean	0	<code>false</code>	Submit CDRs to the backends for processing in batches
<code>size</code>	Unsigned Integer	100	<code>false</code>	The maximum number of CDRs to accumulate before triggering a batch
<code>time</code>	Unsigned Integer	300	<code>false</code>	The maximum time to accumulate CDRs before triggering a batch
<code>scheduleronly</code>	Boolean	0	<code>false</code>	Post batched CDRs on their own thread instead of the scheduler
<code>safeshutdown</code>	Boolean	1	<code>false</code>	Block shutdown of Asterisk until CDRs are submitted

Configuration Option Descriptions

debug

When set to `True`, verbose updates of changes in CDR information will be logged. Note that this is only of use when debugging CDR behavior.

enable

Define whether or not to use CDR logging. Setting this to "no" will override any loading of backend CDR modules. Default is "yes".

unanswered

Define whether or not to log unanswered calls. Setting this to "yes" will report every attempt to ring a phone in dialing attempts, when it was not answered. For example, if you try to dial 3 extensions, and this option is "yes", you will get 3 CDR's, one for each phone that was rung. Some find this information horribly useless. Others find it very valuable. Note, in "yes" mode, you will see one CDR, with one of the call targets on one side, and the originating channel on the other, and then one CDR for each channel attempted. This may seem redundant, but cannot be helped.

In brief, this option controls the reporting of unanswered calls which only have an A party. Calls which get offered to an outgoing line, but are unanswered, are still logged, and that is the intended behavior. (It also results in some B side CDRs being output, as they have the B side channel as their source channel, and no destination channel.)

congestion

Define whether or not to log congested calls. Setting this to "yes" will report each call that fails to complete due to congestion conditions.

endbeforehexten

As each CDR for a channel is finished, its end time is updated and the CDR is finalized. When a channel is hung up and hangup logic is present (in the form of a hangup handler or the `h` extension), a new CDR is generated for the channel. Any statistics are gathered from this new CDR. By enabling this option, no new CDR is created for the dialplan logic that is executed in `h` extensions or attached hangup handler subroutines. The default value is `no`, indicating that a CDR will be generated during hangup logic.

initiatedseconds

Normally, the `billsec` field logged to the CDR backends is simply the end time (hangup time) minus the answer time in seconds. Internally, asterisk stores the time in terms of microseconds and seconds. By setting `initiatedseconds` to `yes`, you can force asterisk to report any seconds that were initiated (a sort of round up method). Technically, this is when the microsecond part of the end time is greater than the microsecond part of the answer time, then the `billsec` time is incremented one second.

batch

Define the CDR batch mode, where instead of posting the CDR at the end of every call, the data will be stored in a buffer to help alleviate load on the asterisk server.



Warning

Use of batch mode may result in data loss after unsafe asterisk termination, i.e., software crash, power failure, kill -9, etc.

size

Define the maximum number of CDRs to accumulate in the buffer before posting them to the backend engines. `batch` must be set to `yes`.

time

Define the maximum time to accumulate CDRs before posting them in a batch to the backend engines. If this time limit is reached, then it will post the records, regardless of the value defined for `size`. `batch` must be set to `yes`.



Note

Time is expressed in seconds.

schedulersonly

The CDR engine uses the internal asterisk scheduler to determine when to post records. Posting can either occur inside the scheduler thread, or a new thread can be spawned for the submission of every batch. For small batches, it might be acceptable to just use the scheduler thread, so set this to `yes`. For large batches, say anything over `size=10`, a new thread is recommended, so set this to `no`.

safeshutdown

When shutting down asterisk, you can block until the CDRs are submitted. If you don't, then data will likely be lost. You can always check the size of the CDR batch buffer with the CLI `cdr status` command. To enable blocking on submission of CDR data during asterisk shutdown, set this to `yes`.

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Configuration_cel

This configuration documentation is for functionality provided by `cel`.

cel.conf

general

Options that apply globally to Channel Event Logging (CEL)

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>enable</code>	Boolean	<code>no</code>	<code>false</code>	Determines whether CEL is enabled
<code>dateformat</code>	String		<code>false</code>	The format to be used for dates when logging
<code>apps</code>	Custom		<code>false</code>	List of apps for CEL to track
<code>events</code>	Custom		<code>false</code>	List of events for CEL to track

Configuration Option Descriptions

apps

A case-insensitive, comma-separated list of applications to track when one or both of APP_START and APP_END events are flagged for tracking

events

A case-sensitive, comma-separated list of event names to track. These event names do not include the leading `AST_CEL`.

- `ALL` - Special value which tracks all events.
- `CHAN_START`
- `CHAN_END`
- `ANSWER`
- `HANGUP`
- `APP_START`
- `APP_END`
- `PARK_START`
- `PARK_END`
- `USER_DEFINED`
- `BRIDGE_ENTER`
- `BRIDGE_EXIT`
- `BLINDTRANSFER`
- `ATTENDEDTRANSFER`
- `PICKUP`
- `FORWARD`
- `LINKEDID_END`
- `LOCAL_OPTIMIZE`

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404375

Asterisk 12 Configuration_chan_motif

Jingle Channel Driver

This configuration documentation is for functionality provided by `chan_motif`.

Overview

Transports

There are three different transports and protocol derivatives supported by `chan_motif`. They are in order of preference: Jingle using ICE-UDP, Google Jingle, and Google-V1.

Jingle as defined in XEP-0166 supports the widest range of features. It is referred to as `ice-udp`. This is the specification that Jingle clients implement.

Google Jingle follows the Jingle specification for signaling but uses a custom transport for media. It is supported by the Google Talk Plug-in in Gmail and by some other Jingle clients. It is referred to as `google` in this file.

Google-V1 is the original Google Talk signaling protocol which uses an initial preliminary version of Jingle. It also uses the same custom transport as Google Jingle for media. It is supported by Google Voice, some other Jingle clients, and the Windows Google Talk client. It is referred to as `google-v1` in this file.

Incoming sessions will automatically switch to the correct transport once it has been determined.

Outgoing sessions are capable of determining if the target is capable of Jingle or a Google transport if the target is in the roster. Unfortunately it is not possible to differentiate between a Google Jingle or Google-V1 capable resource until a session initiate attempt occurs. If a resource is determined to use a Google transport it will initially use Google Jingle but will fall back to Google-V1 if required.

If an outgoing session attempt fails due to failure to support the given transport `chan_motif` will fall back in preference order listed previously until all transports have been exhausted.

Dialing and Resource Selection Strategy

Placing a call through an endpoint can be accomplished using the following dial string:

`Motif/endpoint name/target`

When placing an outgoing call through an endpoint the requested target is searched for in the roster list. If present the first Jingle or Google Jingle capable resource is specifically targeted. Since the capabilities of the resource are known the outgoing session initiation will disregard the configured transport and use the determined one.

If the target is not found in the roster the target will be used as-is and a session will be initiated using the transport specified in this configuration file. If no transport has been specified the endpoint defaults to `ice-udp`.

Video Support

Support for video does not need to be explicitly enabled. Configuring any video codec on your endpoint will automatically enable it.

DTMF

The only supported method for DTMF is RFC2833. This is always enabled on audio streams and negotiated if possible.

Incoming Calls

Incoming calls will first look for the extension matching the name of the endpoint in the configured context. If no such extension exists the call will automatically fall back to the `s` extension.

CallerID

The incoming caller id number is populated with the username of the caller and the name is populated with the full identity of the caller. If you would like to perform authentication or filtering of incoming calls it is recommended that you use these fields to do so.

Outgoing caller id can **not** be set.



Warning

Multiple endpoints using the same connection is **NOT** supported. Doing so may result in broken calls.

motif.conf

endpoint

The configuration for an endpoint.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
context	String	default	false	Default dialplan context that incoming sessions will be routed to
callgroup	Custom		false	A callgroup to assign to this endpoint.
pickupgroup	Custom		false	A pickup group to assign to this endpoint.
language	String		false	The default language for this endpoint.
musicclass	String		false	Default music on hold class for this endpoint.
parkinglot	String		false	Default parking lot for this endpoint.
accountcode	String		false	Account code for CDR purposes
allow	Codec	ulaw,alaw	false	Codecs to allow
disallow	Codec	all	false	Codecs to disallow
connection	Custom		false	Connection to accept traffic on and on which to send traffic out
transport	Custom		false	The transport to use for the endpoint.
maxicecandidates	Unsigned Integer	10	false	Maximum number of ICE candidates to offer
maxpayloads	Unsigned Integer	30	false	Maximum number of payloads to offer

Configuration Option Descriptions

transport

The default outbound transport for this endpoint. Inbound messages are inferred. Allowed transports are `ice-udp`, `google`, or `google-v1`. Note that `chan_motif` will fall back to transport preference order if the transport value chosen here fails.

- `ice-udp` - The Jingle protocol, as defined in XEP 0166.
- `google` - The Google Jingle protocol, which follows the Jingle specification for signaling but uses a custom transport for media.
- `google-v1` - Google-V1 is the original Google Talk signaling protocol which uses an initial preliminary version of Jingle. It also uses the same custom transport as `google` for media.

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Configuration_core

Bucket file API

This configuration documentation is for functionality provided by `core`.

bucket

bucket

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
scheme	String		false	Scheme in use for bucket
created	Custom		false	Time at which the bucket was created
modified	Custom		false	Time at which the bucket was last modified

file

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
scheme	String		false	Scheme in use for file
created	Custom		false	Time at which the file was created
modified	Custom		false	Time at which the file was last modified

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Configuration_features

Features Configuration

This configuration documentation is for functionality provided by `features`.

features.conf

globals

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>featuredigittimeout</code>	Custom	1000	false	Milliseconds allowed between digit presses when entering a feature code.
<code>courtesytone</code>	Custom		false	Sound to play when automon or automixmon is activated
<code>recordingfailsound</code>	Custom		false	Sound to play when automon or automixmon is attempted but fails to start
<code>transferdigittimeout</code>	Custom	3	false	Seconds allowed between digit presses when dialing a transfer destination
<code>atxfernoanswertimeout</code>	Custom	15	false	Seconds to wait for attended transfer destination to answer
<code>atxferdropcall</code>	Custom	0	false	Hang up the call entirely if the attended transfer fails
<code>atxferloopdelay</code>	Custom	10	false	Seconds to wait between attempts to re-dial transfer destination
<code>atxfercallbackretries</code>	Custom	2	false	Number of times to re-attempt dialing a transfer destination
<code>xfersound</code>	Custom	beep	false	Sound to play to during transfer and transfer-like operations.
<code>xferfailsound</code>	Custom	beeperr	false	Sound to play to a transferee when a transfer fails
<code>atxferabort</code>	Custom	*1	false	Digits to dial to abort an attended transfer attempt
<code>atxfercomplete</code>	Custom	*2	false	Digits to dial to complete an attended transfer
<code>atxferthreeway</code>	Custom	*3	false	Digits to dial to change an attended transfer into a three-way call

<code>atxferswap</code>	Custom	*4	false	Digits to dial to toggle who the transferrer is currently bridged to during an attended transfer
<code>pickupexten</code>	Custom	*8	false	Digits used for picking up ringing calls
<code>pickupsound</code>	Custom		false	Sound to play to picker when a call is picked up
<code>pickupfailsound</code>	Custom		false	Sound to play to picker when a call cannot be picked up

Configuration Option Descriptions

atxferdropcall

When this option is set to `no`, then Asterisk will attempt to re-call the transferrer if the call to the transfer target fails. If the call to the transferrer fails, then Asterisk will wait `atxferloopdelay` milliseconds and then attempt to dial the transfer target again. This process will repeat until `atxfercallbackretries` attempts to re-call the transferrer have occurred.

When this option is set to `yes`, then Asterisk will not attempt to re-call the transferrer if the call to the transfer target fails. Asterisk will instead hang up all channels involved in the transfer.

xfersound

This sound will play to the transferrer and transfer target channels when an attended transfer completes. This sound is also played to channels when performing an AMI `Bridge` action.

atxferabort

This option is only available to the transferrer during an attended transfer operation. Aborting a transfer results in the transfer being cancelled and the original parties in the call being re-bridged.

atxfercomplete

This option is only available to the transferrer during an attended transfer operation. Completing the transfer with a DTMF sequence is functionally equivalent to hanging up the transferrer channel during an attended transfer. The result is that the transfer target and transferees are bridged.

atxferthreeway

This option is only available to the transferrer during an attended transfer operation. Pressing this DTMF sequence will result in the transferrer, the transferees, and the transfer target all being in a single bridge together.

atxferswap

This option is only available to the transferrer during an attended transfer operation. Pressing this DTMF sequence will result in the transferrer swapping which party he is bridged with. For instance, if the transferrer is currently bridged with the transfer target, then pressing this DTMF sequence will cause the transferrer to be bridged with the transferees.

pickupexten

In order for the pickup attempt to be successful, the party attempting to pick up the call must either have a `namedpickupgroup` in common with a ringing party's `namedcallgroup` or must have a `pickupgroup` in common with a ringing party's `callgroup`.

featuremap

DTMF options that can be triggered during bridged calls

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>atxfer</code>	Custom		false	DTMF sequence to initiate an attended transfer
<code>blindxfer</code>	Custom	#	false	DTMF sequence to initiate a blind transfer
<code>disconnect</code>	Custom	*	false	DTMF sequence to disconnect the current call
<code>parkcall</code>	Custom		false	DTMF sequence to park a call
<code>automon</code>	Custom		false	DTMF sequence to start or stop monitoring a call
<code>automixmon</code>	Custom		false	DTMF sequence to start or stop mixmonitoring a call

Configuration Option Descriptions

atxfer

The transferee parties will be placed on hold and the transferrer may dial an extension to reach a transfer target. During an attended transfer, the transferrer may consult with the transfer target before completing the transfer. Once the transferrer has hung up or pressed the *atxfercomplete* DTMF sequence, then the transferees and transfer target will be bridged.

blindxfer

The transferee parties will be placed on hold and the transferrer may dial an extension to reach a transfer target. During a blind transfer, as soon as the transfer target is dialed, the transferrer is hung up.

disconnect

Entering this DTMF sequence will cause the bridge to end, no matter the number of parties present

parkcall

The parking lot used to park the call is determined by using either the *PARKINGLOT* channel variable or a configured value on the channel (provided by the channel driver) if the variable is not present. If no configured value on the channel is present, then "default" is used. The call is parked in the next available space in the parking lot.

automon

This will cause the channel that pressed the DTMF sequence to be monitored by the *Monitor* application. The format for the recording is determined by the *TOUCH_MONITOR_FORMAT* channel variable. If this variable is not specified, then *wav* is the default. The filename is constructed in the following manner:

prefix-timestamp-filename

where prefix is either the value of the *TOUCH_MONITOR_PREFIX* channel variable or *auto* if the variable is not set. The timestamp is a UNIX timestamp. The filename is either the value of the *TOUCH_MONITOR* channel variable or the callerID of the channels if the variable is not set.

automixmon

Operation of the *automixmon* is similar to the *automon* feature, with the following exceptions: *TOUCH_MIXMONITOR* is used in place of *TOUCH_MONITOR* *TOUCH_MIXMONITOR_FORMAT* is used in place of *TOUCH_MONITOR_FORMAT* There is no equivalent for *TOUCH_MONITOR_PREFIX*. "auto" is always how the filename begins.

applicationmap

Section for defining custom feature invocations during a call

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>.*</code>	Custom		<code>true</code>	A custom feature to invoke during a bridged call

Configuration Option Descriptions

`.*`

Each item listed here is a comma-separated list of parameters that determine how a feature may be invoked during a call

Example:

```
eggs = *5,self,Playback(hello-world),default
```

This would create a feature called `eggs` that could be invoked during a call by pressing the `*5`. The party that presses the DTMF sequence would then trigger the `Playback` application to play the `hello-world` file. The application invocation would happen on the party that pressed the DTMF sequence since `self` is specified. The other parties in the bridge would hear the `default` music on hold class during the playback.

In addition to the syntax outlined in this documentation, a backwards-compatible alternative is also allowed. The following applicationmap lines are functionally identical:

```
eggs = *5,self,Playback(hello-world),default
```

```
eggs = *5,self,Playback,hello-world,default
```

```
eggs = *5,self,Playback,"hello-world",default
```

featuregroup

Groupings of items from the applicationmap

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>.*</code>	Custom		<code>true</code>	Applicationmap item to place in the feature group

Configuration Option Descriptions

`.*`

Each item here must be a name of an item in the applicationmap. The argument may either be a new DTMF sequence to use for the item or it may be left blank in order to use the DTMF sequence specified in the applicationmap. For example:

```
eggs => *1
```

```
bacon =>
```

would result in the applicationmap items `eggs` and `bacon` being in the featuregroup. The former would have its default DTMF trigger overridden with `*1` and the latter would have the DTMF value specified in the applicationmap.

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402154

Asterisk 12 Configuration_named_acl

This configuration documentation is for functionality provided by `named_acl`.

`named_acl.conf`

`named_acl`

Options for configuring a named ACL

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>permit</code>	ACL		<code>false</code>	An address/subnet from which to allow access
<code>deny</code>	ACL		<code>false</code>	An address/subnet from which to disallow access

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404375

Asterisk 12 Configuration_res_ari

HTTP binding for the Stasis API

This configuration documentation is for functionality provided by `res_ari`.

ari.conf

general

General configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>enabled</code>	Boolean	<code>yes</code>	<code>false</code>	Enable/disable the ARI module
<code>pretty</code>	Custom	<code>no</code>	<code>false</code>	Responses from ARI are formatted to be human readable
<code>auth_realm</code>	String	Asterisk REST Interface	<code>false</code>	Realm to use for authentication. Defaults to Asterisk REST Interface.
<code>allowed_origins</code>	String		<code>false</code>	Comma separated list of allowed origins, for Cross-Origin Resource Sharing. May be set to <code>*</code> to allow all origins.

user

Per-user configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>type</code>	None		<code>false</code>	Define this configuration section as a user.
<code>read_only</code>	Boolean	<code>no</code>	<code>false</code>	When set to yes, user is only authorized for read-only requests
<code>password</code>	String		<code>false</code>	Crypted or plaintext password (see <code>password_format</code>)
<code>password_format</code>	Custom	<code>plain</code>	<code>false</code>	<code>password_format</code> may be set to <code>plain</code> (the default) or <code>crypt</code> . When set to <code>crypt</code> , <code>crypt(3)</code> is used to validate the password. A crypted password can be generated using <code>mkpasswd -m sha-512</code> . When set to <code>plain</code> , the password is in plaintext

Configuration Option Descriptions

type

- `user` - Configure this section as a *user*

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403342

Asterisk 12 Configuration_res_parking

This configuration documentation is for functionality provided by `res_parking`.

res_parking.conf

globals

Options that apply to every parking lot

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>parkeddynamically</code>	Boolean	no	false	Enables dynamically created parkinglots.

parking_lot

Defined parking lots for `res_parking` to use to park calls on

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>context</code>	String	<code>parkedcalls</code>	false	The name of the context where calls are parked and picked up from.
<code>parkext</code>	String		false	Extension to park calls to this parking lot.
<code>parkext_exclusive</code>	Boolean	no	false	If yes, the extension registered as <code>parkext</code> will park exclusively to this parking lot.
<code>parkpos</code>	Custom	701-750	false	Numerical range of parking spaces which can be used to retrieve parked calls.
<code>parkinghints</code>	Boolean	no	false	If yes, this parking lot will add hints automatically for parking spaces.
<code>parkingtime</code>	Unsigned Integer	45	false	Amount of time a call will remain parked before giving up (in seconds).
<code>parkedmusicclass</code>	String		false	Which music class to use for parked calls. They will use the default if unspecified.
<code>comebacktoorigin</code>	Boolean	yes	false	Determines what should be done with the parked channel if no one picks it up before it times out.
<code>comebackdialtime</code>	Unsigned Integer	30	false	Timeout for the Dial extension created to call back the parker when a parked call times out.

<code>comebackcontext</code>	String	<code>parkedcallsttimeout</code>	false	Context where parked calls will enter the PBX on timeout when <code>comebacktoorigin=no</code>
<code>courtesytone</code>	String		false	If the name of a sound file is provided, use this as the courtesy tone
<code>parkedplay</code>	Custom	<code>caller</code>	false	Who we should play the courtesytone to on the pickup of a parked call from this lot
<code>parkedcalltransfers</code>	Custom	<code>no</code>	false	Who to apply the DTMF transfer features to when parked calls are picked up or timeout.
<code>parkedcallreparking</code>	Custom	<code>no</code>	false	Who to apply the DTMF parking feature to when parked calls are picked up or timeout.
<code>parkedcallhangup</code>	Custom	<code>no</code>	false	Who to apply the DTMF hangup feature to when parked calls are picked up or timeout.
<code>parkedcallrecording</code>	Custom	<code>no</code>	false	Who to apply the DTMF MixMonitor recording feature to when parked calls are picked up or timeout.
<code>findslot</code>	Custom	<code>first</code>	false	Rule to use when trying to figure out which parking space a call should be parked with.
<code>courtesytone</code>				If set, the sound set will be played to whomever is set by <code>parkedplay</code>

Configuration Option Descriptions

context

This option is only used if `parkext` is set.

parkext

If this option is used, this extension will automatically be created to place calls into parking lots. In addition, if `parkext_exclusive` is set for this parking lot, the name of the parking lot will be included in the application's arguments so that it only parks to this parking lot. The extension will be created in `context`. Using this option also creates extensions for retrieving parked calls from the parking spaces in the same context.

parkpos

If `parkext` is set, these extensions will automatically be mapped in `context` in order to pick up calls parked to these parking spaces.

comebacktoorigin

Valid Options:

- **yes** - Automatically have the parked channel dial the device that parked the call with dial timeout set by the `parkingtime` option. When the call times out an extension to dial the PARKER will automatically be created in the `park-dial` context with an extension of the flattened parker device name. If the call is not answered, the parked channel that is timing out will continue in the dial plan at that point if there are more priorities in the extension (which won't be the case unless the dialplan deliberately includes such priorities in the `park-dial` context through pattern matching or deliberately written flattened peer extensions).
- **no** - Place the call into the PBX at `comebackcontext` instead. The extension will still be set as the flattened peer name. If an extension the flattened peer name isn't available then it will fall back to the `s` extension. If that also is unavailable it will attempt to fall back to `s@default`. The normal dial extension will still be created in the `park-dial` context with the extension also being the flattened peer name.



Note

Flattened Peer Names - Extensions can not include slash characters since those are used for pattern matching. When a peer name is flattened, slashes become underscores. For example if the parker of a call is called `SIP/0004F2040001` then flattened peer name and therefor the extensions created and used on timeouts will be `SIP_0004F204001`.



Note

When parking times out and the channel returns to the dial plan, the following variables are set:

- `PARKING_SPACE` - extension that the call was parked in prior to timing out.
- `PARKINGSLLOT` - Deprecated. Use `PARKING_SPACE` instead.
- `PARKEDLOT` - name of the lot that the call was parked in prior to timing out.
- `PARKER` - The device that parked the call
- `PARKER_FLAT` - The flat version of `PARKER`

comebackcontext

The extension the call enters will prioritize the flattened peer name in this context. If the flattened peer name extension is unavailable, then the 's' extension in this context will be used. If that also is unavailable, the 's' extension in the 'default' context will be used.

courtesytone

By default, this tone is only played to the caller of a parked call. Who receives the tone can be changed using the `parkedplay` option.

parkedplay

- **no** - Apply to neither side.
- **caller** - Apply only to the call connecting with the call coming out of the parking lot.
- **callee** - Apply only to the call coming out of the parking lot.
- **both** - Apply to both sides.



Note

If courtesy tone is not specified then this option will be ignored.

parkedcalltransfers

- **no** - Apply to neither side.
- **caller** - Apply only to the call connecting with the call coming out of the parking lot.
- **callee** - Apply only to the call coming out of the parking lot.
- **both** - Apply to both sides.

parkedcallreparking

- **no** - Apply to neither side.
- **caller** - Apply only to the call connecting with the call coming out of the parking lot.
- **callee** - Apply only to the call coming out of the parking lot.
- **both** - Apply to both sides.

parkedcallhangup

- `no` - Apply to neither side.
- `caller` - Apply only to the call connecting with the call coming out of the parking lot.
- `callee` - Apply only to the call coming out of the parking lot.
- `both` - Apply to both sides.

parkedcallrecording

- `no` - Apply to neither side.
- `caller` - Apply only to the call connecting with the call coming out of the parking lot.
- `callee` - Apply only to the call coming out of the parking lot.
- `both` - Apply to both sides.

findslot

- `first` - Always try to place in the lowest available space in the parking lot
- `next` - Track the last parking space used and always attempt to use the one immediately after.

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404375

Asterisk 12 Configuration_res_pjsip

SIP Resource using PJProject

This configuration documentation is for functionality provided by `res_pjsip`.

pjsip.conf

endpoint

Endpoint

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>100rel</code>	Custom	yes	false	Allow support for RFC3262 provisional ACK tags
<code>aggregate_mwi</code>	Boolean	yes	false	
<code>allow</code>	Codec		false	Media Codec(s) to allow
<code>aors</code>	String		false	AoR(s) to be used with the endpoint
<code>auth</code>	Custom		false	Authentication Object(s) associated with the endpoint
<code>callerid</code>	Custom		false	CallerID information for the endpoint
<code>callerid_privacy</code>	Custom		false	Default privacy level
<code>callerid_tag</code>	Custom		false	Internal <code>id_tag</code> for the endpoint
<code>context</code>	String	default	false	Dialplan context for inbound sessions
<code>direct_media_glare_mitigation</code>	Custom	none	false	Mitigation of direct media (re)INVITE glare
<code>direct_media_method</code>	Custom	invite	false	Direct Media method type
<code>connected_line_method</code>	Custom	invite	false	Connected line method type
<code>direct_media</code>	Boolean	yes	false	Determines whether media may flow directly between endpoints.
<code>disable_direct_media_on_nat</code>	Boolean	no	false	Disable direct media session refreshes when NAT obstructs the media session
<code>disallow</code>	Codec		false	Media Codec(s) to disallow
<code>dtmf_mode</code>	Custom	rfc4733	false	DTMF mode
<code>media_address</code>	String		false	IP address used in SDP for media handling
<code>force_rport</code>	Boolean	yes	false	Force use of return port
<code>ice_support</code>	Boolean	no	false	Enable the ICE mechanism to help traverse NAT

identify_by	Custom	username	false	Way(s) for Endpoint to be identified
redirect_method	Custom	user	false	How redirects received from an endpoint are handled
mailboxes	String		false	Mailbox(es) to be associated with
moh_suggest	String	default	false	Default Music On Hold class
outbound_auth	Custom		false	Authentication object used for outbound requests
outbound_proxy	String		false	Proxy through which to send requests, a full SIP URI must be provided
rewrite_contact	Boolean	no	false	Allow Contact header to be rewritten with the source IP address-port
rtp_ipv6	Boolean	no	false	Allow use of IPv6 for RTP traffic
rtp_symmetric	Boolean	no	false	Enforce that RTP must be symmetric
send_diversion	Boolean	yes	false	Send the Diversion header, conveying the diversion information to the called user agent
send_pai	Boolean	no	false	Send the P-Asserted-Identity header
send_rpid	Boolean	no	false	Send the Remote-Party-ID header
timers_min_se	Unsigned Integer	90	false	Minimum session timers expiration period
timers	Custom	yes	false	Session timers for SIP packets
timers_sess_expires	Unsigned Integer	1800	false	Maximum session timer expiration period
transport	String		false	Desired transport configuration
trust_id_inbound	Boolean	no	false	Accept identification information received from this endpoint
trust_id_outbound	Boolean	no	false	Send private identification details to the endpoint.
type	None		false	Must be of type 'endpoint'.
use_ptime	Boolean	no	false	Use Endpoint's requested packetisation interval
use_avpf	Boolean	no	false	Determines whether res_pjsip will use and enforce usage of AVPF for this endpoint.

<code>media_encryption</code>	Custom	no	false	Determines whether <code>res_pjsip</code> will use and enforce usage of media encryption for this endpoint.
<code>inband_progress</code>	Boolean	no	false	Determines whether <code>chan_pjsip</code> will indicate ringing using inband progress.
<code>call_group</code>	Custom		false	The numeric pickup groups for a channel.
<code>pickup_group</code>	Custom		false	The numeric pickup groups that a channel can pickup.
<code>named_call_group</code>	Custom		false	The named pickup groups for a channel.
<code>named_pickup_group</code>	Custom		false	The named pickup groups that a channel can pickup.
<code>device_state_busy_at</code>	Unsigned Integer	0	false	The number of in-use channels which will cause busy to be returned as device state
<code>t38_udptl</code>	Boolean	no	false	Whether T.38 UDPTL support is enabled or not
<code>t38_udptl_ec</code>	Custom	none	false	T.38 UDPTL error correction method
<code>t38_udptl_maxdatagram</code>	Unsigned Integer	0	false	T.38 UDPTL maximum datagram size
<code>fax_detect</code>	Boolean	no	false	Whether CNG tone detection is enabled
<code>t38_udptl_nat</code>	Boolean	no	false	Whether NAT support is enabled on UDPTL sessions
<code>t38_udptl_ipv6</code>	Boolean	no	false	Whether IPv6 is used for UDPTL Sessions
<code>tone_zone</code>	String		false	Set which country's indications to use for channels created for this endpoint.
<code>language</code>	String		false	Set the default language to use for channels created for this endpoint.
<code>one_touch_recording</code>	Boolean	no	false	Determines whether one-touch recording is allowed for this endpoint.
<code>record_on_feature</code>	String	automixmon	false	The feature to enact when one-touch recording is turned on.
<code>record_off_feature</code>	String	automixmon	false	The feature to enact when one-touch recording is turned off.

rtp_engine	String	asterisk	false	Name of the RTP engine to use for channels created for this endpoint
allow_transfer	Boolean	yes	false	Determines whether SIP REFER transfers are allowed for this endpoint
sdp_owner	String	-	false	String placed as the username portion of an SDP origin (o=) line.
sdp_session	String	Asterisk	false	String used for the SDP session (s=) line.
tos_audio	Unsigned Integer	0	false	DSCP TOS bits for audio streams
tos_video	Unsigned Integer	0	false	DSCP TOS bits for video streams
cos_audio	Unsigned Integer	0	false	Priority for audio streams
cos_video	Unsigned Integer	0	false	Priority for video streams
allow_subscribe	Boolean	yes	false	Determines if endpoint is allowed to initiate subscriptions with Asterisk.
sub_min_expiry	Unsigned Integer	0	false	The minimum allowed expiry time for subscriptions initiated by the endpoint.
from_user	String		false	Username to use in From header for requests to this endpoint.
mwi_from_user	String		false	Username to use in From header for unsolicited MWI NOTIFYs to this endpoint.
from_domain	String		false	Domain to user in From header for requests to this endpoint.
dtls_verify	Custom		false	Verify that the provided peer certificate is valid
dtls_rekey	Custom		false	Interval at which to renegotiate the TLS session and rekey the SRTP session
dtls_cert_file	Custom		false	Path to certificate file to present to peer
dtls_private_key	Custom		false	Path to private key for certificate file
dtls_cipher	Custom		false	Cipher to use for DTLS negotiation
dtls_ca_file	Custom		false	Path to certificate authority certificate
dtls_ca_path	Custom		false	Path to a directory containing certificate authority certificates

<code>dtls_setup</code>	Custom		false	Whether we are willing to accept connections, connect to the other party, or both.
<code>srtplib_tag_32</code>	Boolean	no	false	Determines whether 32 byte tags should be used instead of 80 byte tags.

Configuration Option Descriptions

100rel

- no
- required
- yes

aggregate_mwi

When enabled, *aggregate_mwi* condenses message waiting notifications from multiple mailboxes into a single NOTIFY. If it is disabled, individual NOTIFYs are sent for each mailbox.

aors

List of comma separated AoRs that the endpoint should be associated with.

auth

This is a comma-delimited list of *auth* sections defined in `pjsip.conf` to be used to verify inbound connection attempts.

Endpoints without an authentication object configured will allow connections without verification.

callerid

Must be in the format Name <Number>, or only <Number>.

callerid_privacy

- allowed_not_screened
- allowed_passed_screened
- allowed_failed_screened
- allowed
- prohib_not_screened
- prohib_passed_screened
- prohib_failed_screened
- prohib
- unavailable

direct_media_glare_mitigation

This setting attempts to avoid creating INVITE glare scenarios by disabling direct media reINVITEs in one direction thereby allowing designated servers (according to this option) to initiate direct media reINVITEs without contention and significantly reducing call setup time.

A more detailed description of how this option functions can be found on the Asterisk wiki <https://wiki.asterisk.org/wiki/display/AST/SIP+Direct+Media+Reinvite+Glare+Avoidance>

- none
- outgoing
- incoming

direct_media_method

Method for setting up Direct Media between endpoints.

- `invite`
- `reinvite` - Alias for the `invite` value.
- `update`

connected_line_method

Method used when updating connected line information.

- `invite`
- `reinvite` - Alias for the `invite` value.
- `update`

dtmf_mode

This setting allows to choose the DTMF mode for endpoint communication.

- `rfc4733` - DTMF is sent out of band of the main audio stream. This supercedes the older **RFC-2833** used within the older `chan_sip`.
- `inband` - DTMF is sent as part of audio stream.
- `info` - DTMF is sent as SIP INFO packets.

media_address

At the time of SDP creation, the IP address defined here will be used as the media address for individual streams in the SDP.



Note

Be aware that the `external_media_address` option, set in Transport configuration, can also affect the final media address used in the SDP.

identify_by

An endpoint can be identified in multiple ways. Currently, the only supported option is `username`, which matches the endpoint based on the username in the From header.



Note

Endpoints can also be identified by IP address; however, that method of identification is not handled by this configuration option. See the documentation for the `identify` configuration section for more details on that method of endpoint identification. If this option is set to `username` and an `identify` configuration section exists for the endpoint, then the endpoint can be identified in multiple ways.

- `username`

redirect_method

When a redirect is received from an endpoint there are multiple ways it can be handled. If this option is set to `user` the user portion of the redirect target is treated as an extension within the dialplan and dialed using a Local channel. If this option is set to `uri_core` the target URI is returned to the dialing application which dials it using the PJSIP channel driver and endpoint originally used. If this option is set to `uri_pjsip` the redirect occurs within `chan_pjsip` itself and is not exposed to the core at all. The `uri_pjsip` option has the benefit of being more efficient and also supporting multiple potential redirect targets. The con is that since redirection occurs within `chan_pjsip` redirecting information is not forwarded and redirection can not be prevented.

- `user`
- `uri_core`
- `uri_pjsip`

rewrite_contact

On inbound SIP messages from this endpoint, the Contact header will be changed to have the source IP address and port. This option does not affect outbound messages sent to this endpoint.

timers_min_se

Minimum session timer expiration period. Time in seconds.

timers

- forced
- no
- required
- yes

timers_sess_expires

Maximum session timer expiration period. Time in seconds.

transport

This will set the desired transport configuration to send SIP data through.



Warning

Not specifying a transport will **DEFAULT** to the first configured transport in `pjsip.conf` which is valid for the URI we are trying to contact.



Warning

Transport configuration is not affected by reloads. In order to change transports, a full Asterisk restart is required

trust_id_inbound

This option determines whether Asterisk will accept identification from the endpoint from headers such as P-Asserted-Identity or Remote-Party-ID header. This option applies both to calls originating from the endpoint and calls originating from Asterisk. If `no`, the configured Caller-ID from `pjsip.conf` will always be used as the identity for the endpoint.

trust_id_outbound

This option determines whether `res_pjsip` will send private identification information to the endpoint. If `no`, private Caller-ID information will not be forwarded to the endpoint. "Private" in this case refers to any method of restricting identification. Example: setting `callerid_privacy` to any `prohib` variation. Example: If `trust_id_inbound` is set to `yes`, the presence of a `Privacy: id` header in a SIP request or response would indicate the identification provided in the request is private.

use_avpf

If set to `yes`, `res_pjsip` will use the AVPF or SAVPF RTP profile for all media offers on outbound calls and media updates and will decline media offers not using the AVPF or SAVPF profile.

If set to `no`, `res_pjsip` will use the AVP or SAVP RTP profile for all media offers on outbound calls and media updates and will decline media offers not using the AVP or SAVP profile.

media_encryption

- `no` - `res_pjsip` will offer no encryption and allow no encryption to be setup.
- `sdes` - `res_pjsip` will offer standard SRTP setup via in-SDP keys. Encrypted SIP transport should be used in conjunction with this option to prevent exposure of media encryption keys.
- `dtls` - `res_pjsip` will offer DTLS-SRTP setup.

inband_progress

If set to `yes`, `chan_pjsip` will send a 183 Session Progress when told to indicate ringing and will immediately start sending ringing as audio.

If set to `no`, `chan_pjsip` will send a 180 Ringing when told to indicate ringing and will NOT send it as audio.

call_group

Can be set to a comma separated list of numbers or ranges between the values of 0-63 (maximum of 64 groups).

pickup_group

Can be set to a comma separated list of numbers or ranges between the values of 0-63 (maximum of 64 groups).

named_call_group

Can be set to a comma separated list of case sensitive strings limited by supported line length.

named_pickup_group

Can be set to a comma separated list of case sensitive strings limited by supported line length.

device_state_busy_at

When the number of in-use channels for the endpoint matches the `devicestate_busy_at` setting the PJSIP channel driver will return busy as the device state instead of in use.

t38_udptl

If set to `yes` T.38 UDPTL support will be enabled, and T.38 negotiation requests will be accepted and relayed.

t38_udptl_ec

- `none` - No error correction should be used.
- `fec` - Forward error correction should be used.
- `redundancy` - Redundancy error correction should be used.

t38_udptl_maxdatagram

This option can be set to override the maximum datagram of a remote endpoint for broken endpoints.

fax_detect

This option can be set to send the session to the fax extension when a CNG tone is detected.

t38_udptl_nat

When enabled the UDPTL stack will send UDPTL packets to the source address of received packets.

t38_udptl_ipv6

When enabled the UDPTL stack will use IPv6.

record_on_feature

When an INFO request for one-touch recording arrives with a Record header set to "on", this feature will be enabled for the channel. The feature designated here can be any built-in or dynamic feature defined in `features.conf`.



Note

This setting has no effect if the endpoint's `one_touch_recording` option is disabled

record_off_feature

When an INFO request for one-touch recording arrives with a Record header set to "off", this feature will be enabled for the channel. The feature designated here can be any built-in or dynamic feature defined in features.conf.



Note

This setting has no effect if the endpoint's one_touch_recording option is disabled

tos_audio

See <https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service> for more information about QoS settings

tos_video

See <https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service> for more information about QoS settings

cos_audio

See <https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service> for more information about QoS settings

cos_video

See <https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service> for more information about QoS settings

dtls_verify

This option only applies if *media_encryption* is set to dtls.

dtls_rekey

This option only applies if *media_encryption* is set to dtls.

If this is not set or the value provided is 0 rekeying will be disabled.

dtls_cert_file

This option only applies if *media_encryption* is set to dtls.

dtls_private_key

This option only applies if *media_encryption* is set to dtls.

dtls_cipher

This option only applies if *media_encryption* is set to dtls.

Many options for acceptable ciphers. See link for more: http://www.openssl.org/docs/apps/ciphers.html#CIPHER_STRINGS

dtls_ca_file

This option only applies if *media_encryption* is set to dtls.

dtls_ca_path

This option only applies if *media_encryption* is set to dtls.

dtls_setup

This option only applies if *media_encryption* is set to *dtls*.

- *active* - *res_pjsip* will make a connection to the peer.
- *passive* - *res_pjsip* will accept connections from the peer.
- *actpass* - *res_pjsip* will offer and accept connections from the peer.

srtp_tag_32

This option only applies if *media_encryption* is set to *sdes* or *dtls*.

auth

Authentication type

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>auth_type</code>	Custom	<code>userpass</code>	<code>false</code>	Authentication type
<code>nonce_lifetime</code>	Unsigned Integer	32	<code>false</code>	Lifetime of a nonce associated with this authentication config.
<code>md5_cred</code>	String		<code>false</code>	MD5 Hash used for authentication.
<code>password</code>	String		<code>false</code>	PlainText password used for authentication.
<code>realm</code>	String		<code>false</code>	SIP realm for endpoint
<code>type</code>	None		<code>false</code>	Must be 'auth'
<code>username</code>	String		<code>false</code>	Username to use for account

Configuration Option Descriptions

auth_type

This option specifies which of the password style config options should be read when trying to authenticate an endpoint inbound request. If set to `userpass` then we'll read from the 'password' option. For `md5` we'll read from 'md5_cred'.

- `md5`
- `userpass`

md5_cred

Only used when `auth_type` is `md5`.

password

Only used when `auth_type` is `userpass`.

domain_alias

Domain Alias

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
-------------	------	---------------	--------------------	-------------

type	None		false	Must be of type 'domain_alias'.
domain	String		false	Domain to be aliased

transport

SIP Transport

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
async_operations	Unsigned Integer	1	false	Number of simultaneous Asynchronous Operations
bind	Custom		false	IP Address and optional port to bind to for this transport
ca_list_file	String		false	File containing a list of certificates to read (TLS ONLY)
cert_file	String		false	Certificate file for endpoint (TLS ONLY)
cipher	Custom		false	Preferred Cryptography Cipher (TLS ONLY)
domain	String		false	Domain the transport comes from
external_media_addresses	String		false	External IP address to use in RTP handling
external_signaling_address	String		false	External address for SIP signalling
external_signaling_port	Unsigned Integer	0	false	External port for SIP signalling
method	Custom		false	Method of SSL transport (TLS ONLY)
local_net	Custom		false	Network to consider local (used for NAT purposes).
password	String		false	Password required for transport
priv_key_file	String		false	Private key file (TLS ONLY)
protocol	Custom	udp	false	Protocol to use for SIP traffic
require_client_cert	Custom		false	Require client certificate (TLS ONLY)
type	None		false	Must be of type 'transport'.
verify_client	Custom		false	Require verification of client certificate (TLS ONLY)
verify_server	Custom		false	Require verification of server certificate (TLS ONLY)

<code>tos</code>	Unsigned Integer	0	false	Enable TOS for the signalling sent over this transport
<code>cos</code>	Unsigned Integer	0	false	Enable COS for the signalling sent over this transport

Configuration Option Descriptions

cipher

Many options for acceptable ciphers see link for more: http://www.openssl.org/docs/apps/ciphers.html#CIPHER_STRINGS

external_media_address

When a request or response is sent out, if the destination of the message is outside the IP network defined in the option `localnet`, and the media address in the SDP is within the localnet network, then the media address in the SDP will be rewritten to the value defined for `external_media_address`.

method

- default
- unspecified
- tlsv1
- sslv2
- sslv3
- sslv23

local_net

This must be in CIDR or dotted decimal format with the IP and mask separated with a slash (/).

protocol

- udp
- tcp
- tls
- ws
- wss

tos

See <https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service> for more information on this parameter.



Note

This option does not apply to the `ws` or the `wss` protocols.

cos

See <https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service> for more information on this parameter.



Note

This option does not apply to the `ws` or the `wss` protocols.

contact

A way of creating an aliased name to a SIP URI

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
type	None		false	Must be of type 'contact'.
uri	String		false	SIP URI to contact peer
expiration_time	Custom		false	Time to keep alive a contact
qualify_frequency	Unsigned Integer	0	false	Interval at which to qualify a contact
outbound_proxy	String		false	Outbound proxy used when sending OPTIONS request

Configuration Option Descriptions

expiration_time

Time to keep alive a contact. String style specification.

qualify_frequency

Interval between attempts to qualify the contact for reachability. If 0 never qualify. Time in seconds.

outbound_proxy

If set the provided URI will be used as the outbound proxy when an OPTIONS request is sent to a contact for qualify purposes.

aor

The configuration for a location of an endpoint

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
contact	Custom		false	Permanent contacts assigned to AoR
default_expiration	Unsigned Integer	3600	false	Default expiration time in seconds for contacts that are dynamically bound to an AoR.
mailboxes	String		false	Mailbox(es) to be associated with
maximum_expiration	Unsigned Integer	7200	false	Maximum time to keep an AoR
max_contacts	Unsigned Integer	0	false	Maximum number of contacts that can bind to an AoR
minimum_expiration	Unsigned Integer	60	false	Minimum keep alive time for an AoR
remove_existing	Boolean	no	false	Determines whether new contacts replace existing ones.

type	None		false	Must be of type 'aor'.
qualify_frequency	Unsigned Integer	0	false	Interval at which to qualify an AoR
authenticate_qualify	Boolean	no	false	Authenticates a qualify request if needed
outbound_proxy	String		false	Outbound proxy used when sending OPTIONS request

Configuration Option Descriptions

contact

Contacts specified will be called whenever referenced by `chan_pjsip`.

Use a separate "contact=" entry for each contact required. Contacts are specified using a SIP URI.

mailboxes

This option applies when an external entity subscribes to an AoR for message waiting indications. The mailboxes specified will be subscribed to. More than one mailbox can be specified with a comma-delimited string.

maximum_expiration

Maximum time to keep a peer with explicit expiration. Time in seconds.

max_contacts

Maximum number of contacts that can associate with this AoR. This value does not affect the number of contacts that can be added with the "contact" option. It only limits contacts added through external interaction, such as registration.



Note

This should be set to 1 and `remove_existing` set to `yes` if you wish to stick with the older `chan_sip` behaviour.

minimum_expiration

Minimum time to keep a peer with an explicit expiration. Time in seconds.

remove_existing

On receiving a new registration to the AoR should it remove the existing contact that was registered against it?



Note

This should be set to `yes` and `max_contacts` set to 1 if you wish to stick with the older `chan_sip` behaviour.

qualify_frequency

Interval between attempts to qualify the AoR for reachability. If 0 never qualify. Time in seconds.

authenticate_qualify

If true and a qualify request receives a challenge or authenticate response authentication is attempted before declaring the contact available.

outbound_proxy

If set the provided URI will be used as the outbound proxy when an OPTIONS request is sent to a contact for qualify purposes.

system

Options that apply to the SIP stack as well as other system-wide settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>timer_t1</code>	Unsigned Integer	500	false	Set transaction timer T1 value (milliseconds).
<code>timer_b</code>	Unsigned Integer	32000	false	Set transaction timer B value (milliseconds).
<code>compact_headers</code>	Boolean	no	false	Use the short forms of common SIP header names.
<code>threadpool_initial_size</code>	Unsigned Integer	0	false	Initial number of threads in the res_pjsip threadpool.
<code>threadpool_auto_increment</code>	Unsigned Integer	5	false	The amount by which the number of threads is incremented when necessary.
<code>threadpool_idle_timeout</code>	Unsigned Integer	60	false	Number of seconds before an idle thread should be disposed of.
<code>threadpool_max_size</code>	Unsigned Integer	0	false	Maximum number of threads in the res_pjsip threadpool. A value of 0 indicates no maximum.
<code>type</code>	None		false	Must be of type 'system'.

Configuration Option Descriptions

timer_t1

Timer T1 is the base for determining how long to wait before retransmitting requests that receive no response when using an unreliable transport (e.g. UDP). For more information on this timer, see RFC 3261, Section 17.1.1.1.

timer_b

Timer B determines the maximum amount of time to wait after sending an INVITE request before terminating the transaction. It is recommended that this be set to 64 * Timer T1, but it may be set higher if desired. For more information on this timer, see RFC 3261, Section 17.1.1.1.

global

Options that apply globally to all SIP communications

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>max_forwards</code>	Unsigned Integer	70	false	Value used in Max-Forwards header for SIP requests.
<code>type</code>	None		false	Must be of type 'global'.

user_agent	String	Asterisk PBX SVN-branch-12-r404375	false	Value used in User-Agent header for SIP requests and Server header for SIP responses.
default_outbound_endpoint	String	default_outbound_endpoint	false	Endpoint to use when sending an outbound request to a URI without a specified endpoint.

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404375

Asterisk 12 Configuration_res_pjsip_acl

SIP ACL module

This configuration documentation is for functionality provided by `res_pjsip_acl`.

Overview

ACL

The ACL module used by `res_pjsip`. This module is independent of `endpoints` and operates on all inbound SIP communication using `res_pjsip`.

There are two main ways of defining your ACL with the options provided. You can use the `permit` and `deny` options which act on **IP** addresses, or the `contactpermit` and `contactdeny` options which act on **Contact header** addresses in incoming REGISTER requests. You can combine the various options to create a mixed ACL.

Additionally, instead of defining an ACL with options, you can reference IP or Contact header ACLs from the file `acl.conf` by using the `acl` or `contactacl` options.

pjsip.conf

acl

Access Control List

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>acl</code>	Custom		false	List of IP ACL section names in <code>acl.conf</code>
<code>contact_acl</code>	Custom		false	List of Contact ACL section names in <code>acl.conf</code>
<code>contact_deny</code>	Custom		false	List of Contact header addresses to deny
<code>contact_permit</code>	Custom		false	List of Contact header addresses to permit
<code>deny</code>	Custom		false	List of IP addresses to deny access from
<code>permit</code>	Custom		false	List of IP addresses to permit access from
<code>type</code>	None		false	Must be of type 'acl'.

Configuration Option Descriptions

acl

This matches sections configured in `acl.conf`. The value is defined as a list of comma-delimited section names.

contact_acl

This matches sections configured in `acl.conf`. The value is defined as a list of comma-delimited section names.

contact_deny

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dotted-decimal notation. Separate the IP address and subnet mask with a slash ('/')

contact_permit

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dotted-decimal notation. Separate the IP address and subnet mask with a slash ('/')

deny

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dotted-decimal notation. Separate the IP address and subnet mask with a slash ('/')

permit

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dotted-decimal notation. Separate the IP address and subnet mask with a slash ('/')

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r403094

Asterisk 12 Configuration_res_pjsip_endpoint_identifier_ip

Module that identifies endpoints via source IP address

This configuration documentation is for functionality provided by `res_pjsip_endpoint_identifier_ip`.

pjsip.conf

identify

Identifies endpoints via source IP address

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
endpoint	String		false	Name of Endpoint
match	Custom		false	IP addresses or networks to match against
type	None		false	Must be of type 'identify'.

Configuration Option Descriptions

match

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dot-decimal notation. Separate the IP address and subnet mask with a slash (/)

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402154

Asterisk 12 Configuration_res_pjsip_notify

Module that supports sending NOTIFY requests to endpoints from external sources

This configuration documentation is for functionality provided by `res_pjsip_notify`.

pjsip_notify.conf

general

Unused, but reserved.

notify

Configuration of a NOTIFY request.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>.*</code>	Custom		<code>true</code>	A key/value pair to add to a NOTIFY request.

Configuration Option Descriptions

`.*`

If the key is `Content`, it will be treated as part of the message body. Otherwise, it will be added as a header in the NOTIFY request.

The following headers are reserved and cannot be specified:

- `Call-ID`
- `Contact`
- `CSeq`
- `To`
- `From`
- `Record-Route`
- `Route`
- `Via`

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402154

Asterisk 12 Configuration_res_pjsip_outbound_registration

SIP resource for outbound registrations

This configuration documentation is for functionality provided by `res_pjsip_outbound_registration`.

Overview

Outbound Registration

This module allows `res_pjsip` to register to other SIP servers.

pjsip.conf

registration

The configuration for outbound registration

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>auth_rejection_permanent</code>	Boolean	yes	false	Determines whether failed authentication challenges are treated as permanent failures.
<code>client_uri</code>	String		false	Client SIP URI used when attempting outbound registration
<code>contact_user</code>	String		false	Contact User to use in request
<code>expiration</code>	Unsigned Integer	3600	false	Expiration time for registrations in seconds
<code>max_retries</code>	Unsigned Integer	10	false	Maximum number of registration attempts.
<code>outbound_auth</code>	Custom		false	Authentication object to be used for outbound registrations.
<code>outbound_proxy</code>	String		false	Outbound Proxy used to send registrations
<code>retry_interval</code>	Unsigned Integer	60	false	Interval in seconds between retries if outbound registration is unsuccessful
<code>forbidden_retry_interval</code>	Unsigned Integer	0	false	Interval used when receiving a 403 Forbidden response.
<code>server_uri</code>	String		false	SIP URI of the server to register against
<code>transport</code>	String		false	Transport used for outbound authentication
<code>type</code>	None		false	Must be of type 'registration'.

Configuration Option Descriptions

auth_rejection_permanent

If this option is enabled and an authentication challenge fails, registration will not be attempted again until the configuration is reloaded.

client_uri

This is the address-of-record for the outbound registration (i.e. the URI in the To header of the REGISTER).

For registration with an ITSP, the client SIP URI may need to consist of an account name or number and the provider's hostname for their registrar, e.g. `client_uri=1234567890@example.com`. This may differ between providers.

For registration to generic registrars, the client SIP URI will depend on networking specifics and configuration of the registrar.

forbidden_retry_interval

If a 403 Forbidden is received, `chan_pjsip` will wait *forbidden_retry_interval* seconds before attempting registration again. If 0 is specified, `chan_pjsip` will not retry after receiving a 403 Forbidden response. Setting this to a non-zero value goes against a "SHOULD NOT" in RFC3261, but can be used to work around buggy registrars.

server_uri

This is the URI at which to find the registrar to send the outbound REGISTER. This URI is used as the request URI of the outbound REGISTER request from Asterisk.

For registration with an ITSP, the setting may often be just the domain of the registrar, e.g. `sip:sip.example.com`.

transport



Note

A *transport* configured in `pjsip.conf`. As with other `res_pjsip` modules, this will use the first available transport of the appropriate type if unconfigured.

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r402154

Asterisk 12 Configuration_res_statsd

Statsd client.

This configuration documentation is for functionality provided by `res_statsd`.

statsd.conf

global

Global configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
enabled	Boolean	no	false	Enable/disable the statsd module
server	IP Address	127.0.0.1	false	Address of the statsd server
prefix	String		false	Prefix to prepend to every metric
add_newline	Boolean	no	false	Append a newline to every event. This is useful if you want to fake out a server using netcat (nc -lu 8125)

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Configuration_res_xmpp

XMPP Messaging

This configuration documentation is for functionality provided by `res_xmpp`.

xmpp.conf

global

Global configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
debug	Custom	no	false	Enable/disable XMPP message debugging
autoprune	Custom	no	false	Auto-remove users from buddy list.
autoregister	Custom	yes	false	Auto-register users from buddy list
collection_nodes	Custom	no	false	Enable support for XEP-0248 for use with distributed device state
pubsub_autocreate	Custom	no	false	Whether or not the PubSub server supports/is using auto-create for nodes
auth_policy	Custom	accept	false	Whether to automatically accept or deny users' subscription requests

Configuration Option Descriptions

autoprune

Auto-remove users from buddy list. Depending on the setup (e.g., using your personal Gtalk account for a test) this could cause loss of the contact list.

client

Configuration options for an XMPP client

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
username	String		false	XMPP username with optional resource
secret	String		false	XMPP password
serverhost	String		false	Route to server, e.g. talk.google.com
statusmessage	String	Online and Available	false	Custom status message
pubsub_node	String		false	Node for publishing events via PubSub

context	String	default	false	Dialplan context to send incoming messages to
priority	Unsigned Integer	1	false	XMPP resource priority
port	Unsigned Integer	5222	false	XMPP server port
timeout	Unsigned Integer	5	false	Timeout in seconds to hold incoming messages
debug	Custom	no	false	Enable debugging
type	Custom	client	false	Connection is either a client or a component
distribute_events	Custom	no	false	Whether or not to distribute events using this connection
usetls	Custom	yes	false	Whether to use TLS for the connection or not
usesasl	Custom	yes	false	Whether to use SASL for the connection or not
forceoldssl	Custom	no	false	Force the use of old-style SSL for the connection
keepalive	Custom	yes	false	If enabled, periodically send an XMPP message from this client with an empty message
autoprune	Custom	no	false	Auto-remove users from buddy list.
autoregister	Custom	yes	false	Auto-register users bfrom buddy list
auth_policy	Custom	accept	false	Whether to automatically accept or deny users' subscription requests
sendtodialplan	Custom	no	false	Send incoming messages into the dialplan
status	Custom	available	false	Default XMPP status for the client
buddy	Custom		false	Manual addition of buddy to list

Configuration Option Descriptions

timeout

Timeout (in seconds) on the message stack. Messages stored longer than this value will be deleted by Asterisk. This option applies to incoming messages only which are intended to be processed by the JABBER_RECEIVE dialplan function.

autoprune

Auto-remove users from buddy list. Depending on the setup (e.g., using your personal Gtalk account for a test) this could cause loss of the contact list.

status

Can be one of the following XMPP statuses:

- chat
- available
- away
- xaway
- dnd

buddy

Manual addition of buddy to the buddy list. For distributed events, these buddies are automatically added in the whitelist as 'owners' of the node(s).

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Configuration_stasis

Stasis message bus configuration.

This configuration documentation is for functionality provided by `stasis`.

`stasis.conf`

`threadpool`

Threadpool configuration.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>initial_size</code>	Integer	0	false	Initial number of threads in the message bus threadpool.
<code>idle_timeout_sec</code>	Integer	20	false	Number of seconds for an idle thread to be disposed of.
<code>max_size</code>	Integer	200	false	Maximum number of threads in the threadpool.

Import Version

This documentation was imported from Asterisk Version Unknown

Asterisk 12 Configuration_udptl

This configuration documentation is for functionality provided by `udptl`.

udptl.conf

global

Global options for configuring UDPTL

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
<code>udptlstart</code>	Unsigned Integer	4000	false	The start of the UDPTL port range
<code>udptlend</code>	Unsigned Integer	4999	false	The end of the UDPTL port range
<code>udptlchecksums</code>	Boolean	yes	false	Whether to enable or disable UDP checksums on UDPTL traffic
<code>udptlfecentries</code>	Unsigned Integer		false	The number of error correction entries in a UDPTL packet
<code>udptlfecspan</code>	Unsigned Integer		false	The span over which parity is calculated for FEC in a UDPTL packet
<code>use_even_ports</code>	Boolean	no	false	Whether to only use even-numbered UDPTL ports
<code>t38faxudpec</code>	Custom		false	Removed
<code>t38faxmaxdatagram</code>	Custom		false	Removed

Import Version

This documentation was imported from Asterisk Version SVN-branch-12-r404375