



KHOMP

[www.khomp.com](http://www.khomp.com)

## Channel Khomp/User Guide

Khomp - Todos os direitos reservados

Última atualização em: 2015-04-15 19:10:10

# Índice

- 1 First considerations
- 2 Configuration
  - 2.1 K3L API Configuration
  - 2.2 Channel Configuration
    - 2.2.1 [globals]
    - 2.2.2 [channels]
    - 2.2.3 [groups]
    - 2.2.4 [cadences]
    - 2.2.5 [fxs-branches]
    - 2.2.6 [fxs-hotlines]
    - 2.2.7 [fxs-options]
    - 2.2.8 **[channels-<string>]**
  - 2.3 Asterisk Configuration
    - 2.3.1 Contexts for E1 channels
    - 2.3.2 Contexts of FXS/FXO/GSM channels
      - 2.3.2.1 Priority settings on the FXS branches
    - 2.3.3 Contexts for SMS messages (GSM only)
    - 2.3.4 Contexts for USSD messages (GSM only)
    - 2.3.5 Contexts for Khomp\_PR channels (KPR)
    - 2.3.6 Transfer contexts
    - 2.3.7 Groups contexts
    - 2.3.8 Using the Dial application
      - 2.3.8.1 Fields relating to the Khomp channel
      - 2.3.8.2 Policy for channel allocation
    - 2.3.9 Grouping channel allocations
      - 2.3.9.1 Cyclical and/or *fair* allocation
      - 2.3.9.2 Allocation by Total Call Time
      - 2.3.9.3 Circular Allocation
      - 2.3.9.4 Available options
    - 2.3.10 List of variables
    - 2.3.11 Description of variables
      - 2.3.11.1 KDropCollectCall
      - 2.3.11.2 KR2Condition
      - 2.3.11.3 KR2GotCategory
      - 2.3.11.4 KR2GotCondition
      - 2.3.11.5 KUserInfoDescriptor
      - 2.3.11.6 KUserInfoData
      - 2.3.11.7 KCallAnswerInfo
      - 2.3.11.8 KOutgoingChannel
      - 2.3.11.9 KTDDStringReceived
    - 2.3.12 CHANNEL() function
- 3 Console commands
  - 3.1 khomp channels disconnect
  - 3.2 khomp calls show
  - 3.3 khomp channels show
  - 3.4 khomp channels statistics
  - 3.5 khomp channels statistics clear
  - 3.6 khomp channels unblock
  - 3.7 khomp dump config
  - 3.8 khomp dump branches
  - 3.9 khomp kommuter
  - 3.10 khomp kommuter count
  - 3.11 khomp links errors
  - 3.12 khomp links reset
  - 3.13 khomp links show
  - 3.14 khomp log console
  - 3.15 khomp log disk
  - 3.16 khomp log rotate

- 3.17 khomp log update
- 3.18 khomp log status
- 3.19 khomp log trace isdn
- 3.20 khomp log trace k3l
- 3.21 khomp log trace r2
- 3.22 khomp modem reset
- 3.23 khomp modem sim card
- 3.24 khomp record
- 3.25 khomp revision
- 3.26 khomp select sim
- 3.27 khomp modem imei
- 3.28 khomp modem imsi
- 3.29 khomp modem iccid
- 3.30 Khomp send command
- 3.31 Khomp send raw command
- 3.32 khomp get
- 3.33 khomp set
- 3.34 khomp sms
- 3.35 khomp summary
- 3.36 khomp ussd
- 4 Additional features
  - 4.1 Applications (applications) and channels
    - 4.1.1 "KUserTransfer" application
    - 4.1.2 "KRecord" application
    - 4.1.3 "KStopRecord" application
    - 4.1.4 Application "KSendSMS"
    - 4.1.5 Application "KSendUSSD"
    - 4.1.6 Application "KEchoCanceller"
    - 4.1.7 Application "KAutoGainControl"
    - 4.1.8 Application "KDTMFSuppression"
    - 4.1.9 Application "KSetVolume"
    - 4.1.10 Application "KAdjustForFax"
    - 4.1.11 Application "KSendFax"
    - 4.1.12 Application "KReceiveFax"
    - 4.1.13 Application "KSelectSimCard"
    - 4.1.14 Aplicação "KSendTDD"
    - 4.1.15 Aplicação "KReceiveTDD"
    - 4.1.16 Channel "Khomp\_SMS"
    - 4.1.17 Channel "Khomp\_USSD"
    - 4.1.18 Channel "Khomp\_PR"
  - 4.2 Management Interface (AMI)
    - 4.2.1 List of commands
      - 4.2.1.1 KSendSMS
      - 4.2.1.2 KSendUSSD
      - 4.2.1.3 KHangup
      - 4.2.1.4 KDialOffHook
    - 4.2.2 Event List
      - 4.2.2.1 Alarm
      - 4.2.2.2 AlarmClear
      - 4.2.2.3 AnswerInfo
      - 4.2.2.4 AntennaLevel
      - 4.2.2.5 OperatorRegistry
      - 4.2.2.6 BranchOffHook
      - 4.2.2.7 BranchOnHook
      - 4.2.2.8 CollectCall
      - 4.2.2.9 NewSMS
      - 4.2.2.10 NewUSSD
      - 4.2.2.11 KDisconnectionCause
  - 4.3 Gateway Interface (AGI)
    - 4.3.1 KSEND SMS
    - 4.3.2 KSENDUSSD
    - 4.3.3 Asterisk Channel Tech API

- 5 Multiparty on KGSM boards
  - 5.1 Prerequisites
  - 5.2 Basic concepts
  - 5.3 Incoming calls
    - 5.3.1 Contexts for additional calls
    - 5.3.2 The *waiting* extension
  - 5.4 New channels
    - 5.4.1 Khomp\_MPTY
    - 5.4.2 Khomp\_Wait
  - 5.5 New applications
    - 5.5.1 KGsmMultipartyStart
    - 5.5.2 KGsmMultipartyStart2
    - 5.5.3 KGsmMultiparty
    - 5.5.4 KGsmMultipartyBreak
    - 5.5.5 KGsmMultipartyOwner
    - 5.5.6 KGsmDial
    - 5.5.7 KGsmHold
  - 5.6 New commands
    - 5.6.1 **khomp calls show**
  - 5.7 New in AMI
    - 5.7.1 New commands
      - 5.7.1.1 KHangup
    - 5.7.2 New events
      - 5.7.2.1 HoldStart
      - 5.7.2.2 HoldStop
      - 5.7.2.3 MptyStart
      - 5.7.2.4 MptyStop
  - 5.8 Some dialplan examples
    - 5.8.1 Simple Conference
    - 5.8.2 Advanced Conference
- 6 Compatibility with DAHDI in Dial String
  - 6.1 Behavior compatible with DAHDI
  - 6.2 Dialstrings examples
- 7 Use of additional patches
- 8 Codes and meanings
  - 8.1 Channel state
  - 8.2 Call state
  - 8.3 Asterisk call states
  - 8.4 GSM Codes
    - 8.4.1 SMS codes (SMS causes)
    - 8.4.2 Call codes (call causes)
    - 8.4.3 General codes (mobile causes)
- 9 Troubleshooting
  - 9.1 Error during installation of kernel module
    - 9.1.1 Compiling the drivers and starting the services
  - 9.2 Setting up special parameters for audio or signaling
  - 9.3 Automatic load of services and kernel modules
- 10 Appendix
  - 10.1 Arrangement of installed files
  - 10.2 Compatibility with Zaptel/DAHDI modules

## First considerations

This document covers information about the *channel* of Khomp as a whole, include the options, *applications*, **CLI** commands, among others.

For first installation procedures, please see the README.

## Configuration

Configuring the Khomp channel is a task that consists of three steps:

- Configuration of the devices through the K3L;
- Setting up the channel;
- Configuration of Asterisk.

These steps are described more fully below.

## K3L API Configuration

In the Channel 4.0 this step is performed in KWebPortal.

In versions prior to 4.0, this step is carried out in **khompwizard** program, a wizard that configures the basic parameters of the system boards. This wizard initializes the configuration files using information from the user when they are needed, initializing the standard settings to default values. Typically, this program runs automatically after installation of the system. However, you may need to run it manually if an update is being performed, or if new cards were added to the system after installing new drivers. If you need to set advanced parameters of the board and/or signaling in versions prior to Channel 4.0, the program **k3lconfig** allows you to access all the available settings for each installed card.

For more information see the documentation of these tools. For solving synchronization issues, check the Troubleshooting section.

## Channel Configuration

The system's default configuration normally meet most user's needs. However, the configuration of the channel Khomp can be modified through the configuration file `'etc/asterisk/khomp.conf'`.

The list of options is as follows:

### [globals]

- **dial-string-like-dahdi**: Enable/disable the dial string compatibility with DAHDI in Dial string form, if enabled all Khomp dial will use the dial string like DAHDI. (

This resource is available since version: **3.1**.

);

- **freepbx-callerid**: Defines if channels should use caller id information defined in the freepbx interface (

This resource is available since version: **4.3**.

);

### [channels]

Define general settings of all channels of Khomp:

- **accountcode**: Sets the default account code to call the channel. This option can be any alphanumeric string (local option);
- **amaflags**: Sets the flag of the Automated Message Accounting standard, affecting the categorization of entries in the Asterisk CDR (local option);
- **audio-packet-length**: Sets the size of packets sent by the audio channel for Asterisk, in bytes (the default value of this option is 128 bytes) (

This resource is available since version: **3.0**.

);

- **audio-rx-sync**: Sets the synchronization system to be used in the treatment of audio received by the board, it can be adjusted for the following options. (the default value is "auto"): (This feature is **deprecated** since version **4.0**.);
  - **board-sync**: For each new packet of incoming audio, reports a new packet for Asterisk. Available if 'audio-packet-length' is same size as board packet length currently 128 bytes);

- **softtimer-kernel**: Available for kernel 2.6.22+ and {g, eg}libc 2.7+, uses timerfd syscalls for creating timers which synchronize the audio packets reported to Asterisk;
- **softtimer-thread**: Creates a thread which signals all channels for pending data each (audio-packet-length/8) milliseconds;
- **auto**: Selects 'board-sync' synchronization method if available, falling back 'softtimer-kernel' if there is system support, and choosing 'softtimer-thread' if all the above fail;
- **auto-fax-adjustment**: Enable ("yes") or disables ("no") the automatic adjustment of the channel (disable the echo canceller and the suppression DTMF) tone to detect FAX (local option) ;
- **auto-gain-control**: Enable ("yes") or disables ("no") the activation of the automatic gain control (AGC) by the channel (local option);
- **callgroup**: Sets the default groups call on all channels (local option);
- **context-digital**: Context for incoming connections on digital devices (the default is "khomp-DD-LL", where "DD" will be replaced at the time of connection by the device number, "LL" by the number of the link, "CCC" by channel number and "SSSS" for the device serial number);
- **context-fxo**: Context for incoming connections on FXO devices (the default is "khomp-CC-DD", where "DD" will be replaced at the time of connection by the device number, "CC" by channel number, and "SSSS" by the device serial number);
- **context-fxo-alt**: Alternative input context for FXO devices (the default is empty - when set, follows the same substitution rules for **context-fxo**);
- **context-fxs**: Context for incoming connections on FXS devices (the default is "khomp-CC-DD", where "DD" will be replaced at the time of connection by the device number, "CC" by channel number, and "SSSS" by the device serial number);
- **context-fxs-alt**: Alternative input context for FXS devices (the default is empty - when set, follows the same substitution rules for **context-fxs**);
- **context-gsm-call** (or **context-gsm**): Context of entry for GSM devices (the default is "khomp-CC-DD", where "DD" will be replaced at the time of connection by the device number, "CC" by channel number, and "SSSS" by the device serial number);
- **context-gsm-alt-call** (or **context-gsm-alt**): Alternative input context for GSM (the default is zero - when set, follows the same substitution rules for **context-gsm-call**);
- **context-gsm-sms**: Context for incoming SMSs (the default is "khomp-sms-CC-DD", where "DD" will be replaced by the number of device, "CC" by channel number and "SSSS" by the device's serial number);
- **context-gsm-wait**: Context used for pre-processing of incoming GSM calls that are on waiting state - to disable this feature, use **none** (the default is "khomp-wait-CC-DD", where "DD" will be replaced by the number of device, "CC" by channel number and "SSSS" by the device's serial number); (This feature is **deprecated** since version **4.0.**);
- **context-pr**: Context for incoming connections on devices KPR (default is "khomp-CC-DD", where "DD" will be replaced at the time of connection by the device number, "CC" by channel number);
- **context**: Context of entry for all channels of technology **Khomp** (local option);
- **delay-ringback-co**: Sets the delay to enable the generation of call control tone (ringback) by the channel Khomp when there is a ringback indication from signaling and there is no audio being sent by the channel which indicated the situation (local option);
- **delay-ringback-pbx**: Sets the delay to enable the generation of call control tone (ringback) by the channel Khomp when there is a ringback indication, and the audio has no tone (silence) (local option);
- **disconnect-delay**: Sets the time in milliseconds to perform processing a disconnect event, to ignore situations where other equipment performing the double service to overthrow collect calls (local option);
- **drop-collect-call**: Enables/Disables the action of dropping collect calls. If enabled, all collect calls will be dropped no matter what KDropCollectCall is set to (the default value is "no") (local option);
- **echo-canceller** (former 'echocanceller'): Active ("yes") or disables ("no") echo cancellation automatic channel (local option);
- **enabled**: Defines if channels should be enabled or not. Useful in specific channel sections. Use for debugging purposes only, not recommended for production environments. (local option) (

This resource is available since version: **3.0.**

);

- **fxo-send-pre-audio**: When enabled ("yes") releases audio channel before the outgoing call connection devices KFXO (the default value is "yes");
- **fxo-fsk-detection**: Defines the kind of **CallerID** detection, which enables recognition of **FSK** or **DTMF** tones for originator number communication from different standards.  
**NOTE**: The originator number detection using **BINA DTMF** standard is made by the device DSPs, being always active, and is not affected by this configuration.

For more information about the available options, please check the configuration file examples (local option) (

This resource is available since version: **3.0**.

- **fxo-fsk-timeout**: Defines the maximum time for recognizing the originator number using the **CallerId** feature (default value is 2000 ms) (local option) (

This resource is available since version: **3.0**.

- **fxs-digit-timeout**: Defines the timeout, in seconds, between digits of a FXS device's extension (local option);
- **fxs-global-orig**: Start number for sequential branch numbering in FXS devices that are not listed in the **[fxs-branches]** section (the numbering follows ascending order from device number and physical channel number) (default is "0");
- **fxs-co-dialtone**: Sequences of numbers, separated by commas, which fires a continuous tone (of central office) in FXS branches (eg: "0,99" means that, when you dial "0" or "99", the user will hear a continuous dial tone) (default is empty);
- **fxs-send-bina-dtmf** (**fxs-bina** until version **3.0 001**): When enabled ("yes"), calls to FXS lines will send digits corresponding to the source phone identification using **BINA DTMF** signaling (the default value is "yes") (local option);
- **fxs-send-fsk**: When enabled ("yes"), sends the originator number using FSK signaling to calls made on FXS branches (default value is "yes") (local option)

This resource is available since version: **3.0**.

- **fxs-sharp-dial**: Enables/disables the use of sharp (#) as an end-of-number digit on FXS channels for instant dialing. This does not affect special numbers which start on sharp, like #8 or #1. (the default value is "no") (local option); (

This resource is available since version: **3.0**.

- **has-ctbus**: Enable ("yes") or disables ("no") support for CT-bus on **CTI** boards (not available on **SPX** models) (local option); (This feature is **deprecated** since version **4.0**.);
- **ignore-letter-dtmfs**: Defines if the channel should ignore some uncommon DTMF digits detected by the device (A, B, C and D). However, if you need to pass those digits through the device, you may need to set this option to 'no' (the default value is "yes") (local option);
- **input-volume**: Sets the volume gain for incoming audio (entering the device), from -10 to +10 (local option);
- **kommutter-activation**: Sets whether to activate devices kommuter found in the system will be done automatically ("auto") by the channel, or manually ("manual") by the user through the command "khomp kommuter on/off" (

This resource is available since version: **2.4.1**.

- **kommutter-timeout**: Sets the timeout (in seconds) for initializing the kommuter devices. If this timeout is reached without receiving notification of the channel, the devices will switch back to "off" condition. The minimum value is "0", where the links will always remain switched "on", and the maximum is "255" (

This resource is available since version: **2.4.1**.

- **load-error**: Defines the behaviour of the module load function if communication to Khomp devices and/or devices cannot be started. Possible values: failure - Return failure and halts Asterisk execution; skip - Skips loading this module; (the default value is "failure");
- **language**: Set language to Khomp channel calls (local option);
- **log-to-console**: Set log messages to be printed on the console;
- **log-to-disk** (old "log"): Set log messages to be saved to disk;
- **mohclass**: Sets class of music-on-hold for Khomp channel calls (local option);
- **native-bridge** (old "bridge"): Activate ("yes") or disables ("no") native bridged mode, where the audio is switched

directly within the same board - which happens when the two bridged channels are on the same board and no Asterisk option limits the direct switching (the default value is "yes") (local option); (This feature is **deprecated** since version **4.0.**);

- **optimize-audio-path**: Sets whether the channel should try to eliminate the audio delay eliminating **aggressively** audio packets that are not readily handled by Asterisk. This ensures minimum delay the audio to the user and avoids delays associated with SIP clients poorly programmed. However, depending on the scheduling policy of the system, this can result in excessive discarding of packets and jittered audio.  
**This option should not be changed naively!** (default is "no") (local option).
- **out-of-band-DTMF** (former **dtmfsuppression**): Activate ("yes") or disables ("no") the removal and DTMF sending these out-of-band (local option);
- **output-volume** (formerly the "volume"): Sets the output volume of leads. Ranges from -10 to +10 (local option);
- **pickupgroup**: Sets the default group that can pull calls being received (local option);
- **pulse-forwarding** (former 'pulsedetection'): Active ("yes") or disables ("no") for the detection of pulses and converting them into DTMF (local option);
- **r2-preconnect-wait** (former 'r2preconnectwait'): Sets the timeout sending the ringback signaling, protocol R2/MFC to start sending audio silence. Only used when "r2-strict-Behavior" is set to "no" (local option);
- **r2-strict-Behaviour**: Enable ("yes") or disables ("no") the behavior of signaling R2/MFC as the standard sets. The default is "no", and can be changed to "yes" if needed to receive / send data precise signaling protocol (condition B, for example) (local option);
- **record-prefix**: Sets the base path for the files recording the links. If not specified, the default path used is "/var/spool/asterisk/monitor/" (local option); (This feature is **deprecated** since version **4.0.**);
- **recording**: Sets whether automatic recording should occur (local option); (This feature is **deprecated** since version **4.0.**);
- **setvar**: Assigns a value to a variable that is activated when the channel is created and is automatically destroyed when the channel is hung up. Usage: setvar=VARIABLENAME=VALUE. You can have multiple *setvars* to set more than one variable (

This resource is available since version: **4.3.1.**

);

- **suppression-delay** (former **suppressiondelay**): Activate ("yes") or disables ("no") the delay necessary to suppress DTMF. If disabled ("no"), also disables suppression of DTMF (local option);
- **trace**: Set debugging options. Should not be used in production unless absolutely necessary;
- **user-transfer-enable**: Enable or disables the channel to transfer between Asterisk® and another PBX (via user signaling, QSig or FXO FLASH) (

This resource is available since version: **3.0.**

) (opção local);

- **user-transfer-digits**: Defines a sequence of DTMF digits to initiate the transfer between Asterisk® and another PBX (using user signaling, like QSig or FXO FLASH) (

This resource is available since version: **2.4.1.**

);

- **qsig-transfer-facility**: Defines the **facility** to be used by the ISDN protocol for transfers between PBXs (for more info, please check the examples in the configuration file) (local option) (

This resource is available since version: **3.0.**

);

- **flash-behaviour**: Defines the flash behaviour. Possible values: **pendulum**, **xfer** (attended transfer) and **auto** (local option) (

This resource is available since version: **3.0.**

);

- **pendulum-digits**: Defines the DTMF digit sequence used to answer a waiting call (placing the current on hold), place the active call on hold and start a new outgoing call, and to alternate between answered calls (local option) (



This resource is available since version: **3.0**.

);

- **pendulum-native**: Defines the channels where the pendulum feature for outgoing will be enabled automatically. Possible values: **none** (no channel will have the feature by default), **fxs** (enable the feature for FXS channels only) and **all** (all channels will have the feature) (local option) (

This resource is available since version: **3.0**.

);

- **pendulum-incoming**: Defines the channels where the pendulum feature for incoming will be enabled automatically. Possible values: **none** (no channel will have the feature by default), **fxs** (enable the feature for FXS channels only) and **all** (all channels will have the feature) (local option) (

This resource is available since version: **3.0**.

);

- **pendulum-timeout**: Defines the timeout for an waiting call to stay in the ringing state without being answered. After this, the waiting call is dropped, and the waiting cadence is stopped (default = 20000ms) (local option) (

This resource is available since version: **3.0**.

);

**NOTE:** Options marked as "local option" can be used in specific settings per device/link/channel/group, for details see information about the section **[channels-<string>]**

This resource is available since version: **3.0**.

.

## [groups]

Defines the groups to be used in channel allocation.

In this case, the options are used to define names for strings *allocationchannels*. The format follows the standard **<group name> = <allocation string>**, where the **allocation string** is the same string used in the Dial application, and **group name** is an arbitrary name chosen by the user.

For example, to define the group **pstn** as the channels 0 and 5 of the device 0, the following line could be used:

```
pstn = b0c0 + b0c5
```

This group, in turn, could be used in the application **Dial as Dial (Khomp/Gpstn/...)**'.

You can associate a given input context to a channel group, simply specify a name of context string after the allocation, separated by ':'.

For example, to define the same group **pstn** as channels 0 to 20 of device 0, and defining the incoming context to for channels in this groups to **from-pstn**, one could use the line:

```
pstn = b0c0-20:from-pstn
```

This group would be used the same way as before in the **Dial** application, and all the calls coming from these channels would be treated in context **from-pstn**.

## [cadences]

Defines settings for the channel cadences.

In this case, the options are names cadences, followed by one or two pairs of numbers - that define the ranges of tone and silence to be used in cadences.

For details, please refer to the configuration file for examples.

### [fxs-branches]

Defines source numbers for the device KFXS.

In this case, the options are sequences of prefixes of branches and serial numbers of the devices, which define the basic numbers of source addresses, and the numerical order of the devices. The format of the options is:

```
prefix = serial1, serial2, ....
```

For example, if two **KFXS-300 SPX** devices with serial numbers **K0374** and **K2352** must be numbered sequentially, starting from branch 200, you may write:

```
200 = 374, 2352
```

This will define the first branch of device 'K0374 as number 200, the second as 201, and so on. The first branch from device K2352 will have number 230 (as K0374 has 30 channels), the second will be numbered 231, and so on - until the last channel, numbered 259.

For more details, please refer to the configuration file for other examples.

### [fxs-hotlines]

Sets hotlines for the KFXS based devices

In this case, the options are sequences of branches and sequences of destination numbers, which define branches to be treated as "hotlines" and numbers to be dialed when they are take off hook. For instance:

```
100 = 1234  
200 = 4321
```

In the first line, the branch numbered 100 will call extension 1234 when taken off hook, while in the second one, branch 200 will call number 4321 when taken off hook.

### [fxs-options]

Allows you to set specific settings for FXS extension.

In this case, the settings are extension numbers (based on those defined in the **[fxs-branches]** section or by the option **fxs-global-orig**), and the options and their values.

The options available are:

- pickupgroup;
- callgroup;
- context;
- input-volume;
- output-volume;
- language;
- mohclass;
- amaflags;
- accountcode;
- calleridnum;
- calleridname;
- mailbox;
- setvar (

This resource is available since version: **4.3.1**.

).

- namedcallgroup (

This resource is available since version: **4.3.1**.

).

- namedpickupgroup (

This resource is available since version: **4.3.1**.

).

Each option is separated from each other by a pipe "|" or a slash "/" and defined after the colon ":". Example:

```
200 = language:en|context:master-branch
```

For more information on the syntax and examples, please refer to the configuration file.

- **NOTE:** Values defined by this section will overwrite values defined by section **[channels-<string>]** below.

### **[channels-<string>]**

This resource is available since version: **3.0**.

Allows to set specific configurations for groups of channels, where **<string>** follows the same format as used by the allocation string of application **Dial**.

Examples:

```
[channels-Gpstn]
echo-canceller = no
auto-fax-adjustment = no
```

```
[channels-b010+b110]
echo-canceller = no
auto-fax-adjustment = yes
```

The options available in these sections are described in the section above **[channels ]**, marked also as "local options".

For more information visit the configuration file '**khomp.conf**'.

## **Asterisk Configuration**

When connections are received on the Khomp devices, they are forwarded by the channel to specific contexts within the dialplan of Asterisk®. These settings can be changed via the configuration file **khomp.conf**, available on the Asterisk configuration directory (by default, "/etc/asterisk").

For details about the specific contexts, see section Channel configuration.

Below are details of how to configure the settings for incoming calls:

### **Contexts for E1 channels**

For E1 devices, inbound contexts are predefined as option **context-digital** with default value as following:

```
context-digital = khomp-DD-LL
```

This standard defines the context that links will be redirected in accordance with the number of the device and number of the link: **DD** is the device number (two digits), and **LL** is the number of the link (also with two digits).

However, it is possible to configure other inbound contexts, with different formats. There is format **CCC**, which means the channel number on the device (three digits), and **SSSS**, which represents the serial device number (with four digits).

## Examples for configuration entries:

```
; Sequential device number and the link
context-digital = khomp-DD-LL      (eg: khomp-01-00)

; Serial device number and sequential link
context-digital = khomp-SSSS-LL    (eg: khomp-3049-00)

; Sequential device number and the channel
context-digital = khomp-DD-CCC     (eg: khomp-00-001)

; Receive all calls in one context
context-digital = khomp-digital
```

## Follows an example the context usage inside the dialplan:

```
; The present context in 'extensions.conf' will handle calls
; that come from the link 0 (first link) of the device 0.
;
[khomp-00-00]
```

## Another example, using the same format:

```
; The present context in 'extensions.conf' will handle calls
; that come from the link 1 (second link) of the device 0.
;
[khomp-00-01]
```

## A complete example, with a few simple actions:

```
[Khomp-00-00]
exten => 1234,1,Dial(Khomp/b0L1/2345)

exten => _23XX,1,Dial(SIP/11${EXTEN:2})

[Khomp-00-01]
exten => 1111,1,Dial(Khomp/b0L0/2345)
```

## This dialplan defines that:

1. The incoming calls on the link **0** of the device **0** will have the following handling:
  - Calls to the extension 1234 for will be redirected to the second link on the first device (b0L1), calling number **2345**;
  - Calls to any four digit number starting with **23** will be redirected to SIP phones numbered **11** plus the last two digits of the number dialed.
2. The incoming calls on the link **1** of the device **0** for the number **1111** will be redirected to the first link of the first device (b0L0) calling number **2345**.

## Contexts of FXS/FXO/GSM channels

Inbound calls are routed by the channel same way as in E1 channels, using the predefined contexts below:

```
context-gsm = khomp-DD-CC; GSM devices

context-fxs = khomp-DD-CC; FXS devices

context-fxo = khomp-DD-CC; FXO devices
```

There is also an alternative context, which has the following default value:

```
context-gsm-alt = khomp-DD; GSM devices

context-fxs-alt = khomp-DD; FXS devices

context-fxo-alt = khomp-DD; FXO devices
```

For these options, **DD** is the device number (two digits), and **CC** is the channel number (also two digits). There is also the **SSSS** format, which represents the serial number device.

- **NOTE:** In the **KGSM** device, incoming calls are **always** redirected to the "**s**" extension, since the GSM protocol does not identify the target number, only the originator - if not omitted.

## Priority settings on the FXS branches

On calls originated from an FXS branch, the channel driver searches for a valid extension in three different contexts. The priority of contexts is as follows:

1. Specific context, defined in the following way (the last configured one will be used):
  - Branch-specific extension, defined in section **[fxs-options]** of file **khomp.conf**;
  - Channel-specific extension, defined in section **[channels]** or **[channels-<string>]**.
2. Context defined in **context-fxs**;
3. Context defined in **context-fxs-alt**.

If no valid extension can be found, the extension is searched in the contexts above, and also in the **default** context. If the dialing timeout is reached (option **fxs-digit-timeout**), the search will consider all digits dialed until the timeout; if no extension is found, it will search extension "t"; if nothing is found again, then extension "i" will be searched; if no extension is found, "fast-busy" cadence will be played.

## Contexts for SMS messages (GSM only)

SMS messages are received by the Khomp channel and forwarded to Asterisk as a normal connection with no audio. This connection has some variables set with information received in the message - for more information on these variables, see the channel variables documentation. This context can also be modified in the same way as the above contexts.

The default value for this option follows:

```
context-gsm-sms = khomp-sms-DD-CC
```

Where **DD** is the device number (two digits), and **CC** is the channel number (also with two digits). For example:

```
[Khomp-sms-00-01]
exten => s,1,System(/usr/bin/handleSMS.sh ${KSmsFrom} ${KSmsBody})
```

**Warning:** if an SMS context is added to the dialplan while Asterisk is running, it is necessary to execute the following CLI commands: *dialplan reload* and *module reload chan\_khomp.so*. This command sequence indicates the channel that a new SMS context may be available.

## Contexts for USSD messages (GSM only)

USSD messages are received by the Khomp channel and forwarded to Asterisk as a normal connection with no audio. This connection has some variables set with information received in the message - for more information on these variables, see the channel variables documentation. This context can also be modified in the same way as the above contexts.

The default value for this option follows:

```
context-gsm-ussd = khomp-ussd-DD-CC
```

Where **DD** is the device number (two digits), and **CC** is the channel number (also with two digits). For example:

```
[Khomp-ussd-00-01]
exten => s,1,System(/usr/bin/handleUSSD.sh ${KUssdMessage})
```

**Warning:** if a USSD context is added to the dialplan while Asterisk is running, it is necessary to execute the following CLI commands: *dialplan reload* and *module reload chan\_khomp.so*. This command sequence indicates the channel that a new USSD context may be available.

## Contexts for Khomp\_PR channels (KPR)

For these devices, inbound links have a pre-defined context, as shown below:

Khomp Ind. e Com. Ltda. 1996-2015 :: Rua Joe Collaco, 253 :: Florianopolis/SC :: Brasil :: +55 48 3722-2900

```
context-pr = khomp-CC-DD
```

In this case, **DD** is the device number (two digits), and **CC** is the channel number (also two digits). The name and format of this context can also be changed through the "context-pr" in the configuration file.

## Transfer contexts

Asterisk is responsible for initiating transfers, buffer the digits and perform all other steps in the logic of a call transfer. However, each channel allocated in Asterisk (either for an incoming or outgoing call) lets you specify only one context for transfer. This implies that only one context (besides the default) can be used during a call transfer.

In this case, the context chosen by the channel driver follows the following rule for the channel to make the transfer:

- If the channel **originated** the call, the context chosen is the same context that was used to originate the call;
- If the channel **received** the call, the context chosen is the same where the call was received.

## Groups contexts

If you want to define specific settings for certain groups of channels, this can be accomplished using the section **groups**, in the configuration file **khomp.conf**.

This section is detailed in the section Channel Configuration.

## Using the Dial application

The application **Dial** is responsible for generating calls from the Asterisk® dialplan. This application can be used to generate calls from different channel technologies, following a specific format to define destination, dialing options and define the communication channel to be used.

### Fields relating to the Khomp channel

When used for **Khomp** channels, the dial string can have two, three or four fields separated by slash (/). Some example strings:

```
Dial (Khomp/B2L0/32625644,30,tT)
Dial (Khomp/*B2L0/32625644)
Dial (Khomp/S0411/99991234)
Dial (Khomp/Gpstn/99991234)
Dial (Khomp/*Gpstn/99991234,,t)
Dial (Khomp/B2C58/32625644/category=4:orig=4855553232,,tT)
Dial (Khomp/b0c9,30)
Dial (Khomp/b0c1+b0c14)
Dial (Khomp/r304)
```

In the first five examples, three fields have been specified; in the fourth, four fields are used; in the last three examples, just two are used.

**NOTE:** The values separated by comma (,) are options for the application **Dial**, not for the Khomp channel.

The fields description for the Khomp channel:

- **1st field: Khomp:** identifying the type of *channel* in question;
- **2nd field: B2L0, S0411, Gpstn**, etc: represents the **Policy for channel allocation** (detailed below);
- **3rd field: 32625644 and 99991234:** the destination numbers (missing for calls to **KFXS** channels);
- **4th field: category=4:orig=4855553232:** additional options, detailed below.

NOTE: The **Dial** allocation string with only two fields is specific to the KFXS devices, where the destination is the channel itself.

### Policy for channel allocation

The policy for allocation of channels on the **Khomp** channel driver can be specified in the **Dial** string itself or in the **[groups]** section (inside the configuration file **khomp.conf**). To specify devices, links and channels, the following syntax is

available (considering X, Y and Z as any numbers):

- **bX** -- search the channels on the device **X**, ascending order;
- **bXLY** -- search channel on the link **Y** from **X** device, ascending order;
- **bXcY** -- try to allocate channel **Y** from device **X**;
- **bXcY-Z** -- search for channels starting from channel **Y** to channel **Z** (inclusive) of device **X**, ascending order;
- **BXcY-Z** -- same as above, but descending order;
- **sX** -- search the channels on the device of serial number **X**, ascending order;
- **sXLY** -- search channel on the link **Y** from device of serial number **X**, ascending order;
- **sXcY** -- try to allocate channel **Y** from device of serial number **X**;
- **sXcY-Z** -- search for channels starting from channel **Y** to channel **Z** (inclusive) from device of serial number **X**, ascending order;
- **SXcY-Z** -- same as above, but descending order.

To search for extensions of **KFXS** devices according to the extension number, can be used the following syntax (whereas X and Y valid extension numbers):

- **rX**- search branch numbered **X**;
- **RX**- equivalent to the above;
- **rX-Y**- search from branch **X** to **Y**, ascending order;
- **RX-Y**- search from branch **X** to **Y**, descending order.

The capitalization of the letter 'B', 'S' or 'R' defines the search order of the channels: if capitalized, order is descending; otherwise, ascending.

As for the allocation of channels across groups, the following syntax is available:

- **ggroupname** - uses the string defined the group "groupname" in the configuration file (detailed in the configuration section of the channel).
- **Ggroupname** - equivalent to the above.

## Grouping channel allocations

There are cases where you need to get more channels for a particular device or particular group of extensions. For this, there is an extension available in string allocation, with respect to the use of token sum (+) to concatenate multiple strings *binding*, as follows:

```
Dial (Khomp/B1L0+B2L0/32332933,30,tT)
Dial (Khomp/*B2+B3+B4/99887766)
Dial (Khomp/S0411+B1L0/99887766)
Dial (Khomp/Gpstn1+Gpstn2/99991234)
Dial (Khomp/*gOperadoral+gOperadora2/98891234)
Dial (Khomp/OgOperadoral+gOperadora2/98891234)
Dial (Khomp/agOperadoral/98891234)
```

This grouping is available for the *application Dial* and on the specification of groups. The processing of allocation takes place from left to right - except when using cyclic channel allocation, where **all** the specified channels are scanned simultaneously.

## Cyclical and/or fair allocation

Another way for allocation of channels is the cyclic and/or *fair* allocation, which chooses the channel that has **completed** the the lowest number of **outgoing** calls. This mode of allocation may be used by passing an asterisk (\*) before the allocation string of channels (as can be seen in the section above, in the second and fifth examples).

When started with an asterisk (\*), other forms of allocation (increasing, decreasing, etc) are used to decide what channel will be allocated when there are two or more channels with less number of outgoing calls.

- **WARNING: The use of fair and/or cyclic is recommended only for analog (KFXO), branches (KFXS) and cellular interface (KGSM) devices.** E1 connections should allocate channels in one way (ascending/descending) from one side and the opposite on the other to avoid problems of double seizure (which may occur in R2/MFC signaling).

Fair/cyclic allocation also costs more in memory and processor footprint, which tends to be a higher cost in E1 due to the higher number of channels (30 in each link).

For these reasons, fair/cyclic allocations should only be used on signalizations where it can represent any real

difference, like equalizing the charge costs of the lines, the total usage, or the number of connections received by each branch.

Since version 4.3 the channel uses a cache to improve the performance of cyclic allocations. Depending on the configuration used, the balance between channels can be distorted at the beginning of the allocation process, but over time it tends to stabilize.

## Allocation by Total Call Time

It is also possible to allocate channels by choosing the channel that has the lowest total time for outgoing calls (

This resource is available since version: **4.3.0**.

). This mode of allocation may be used by passing the letter **O** (uppercase or lowercase) before the allocation string of channels (as can be seen in the section above, in the penultimate example).

## Circular Allocation

Always tries to allocate the subsequent channel to the one used in the last allocation (

This resource is available since version: **4.3.1**.

). This mode of allocation may be used by passing the letter **A** (uppercase or lowercase) before the allocation string of channels (as can be seen in the section above, in the last example).

## Available options

- **answer\_info**: When specified (take no parameters), report answer information to the user through AMI event **AnswerInfo** and variable **KCallAnswerInfo** (the later needing a patch to **Asterisk®**);
- **category**: When set to a numeric value, sets the category of outgoing call to this value (available only in R2/MFC signaling);
- **drop\_on**: When set to "message\_box", "human\_answer", "answering\_machine", "carrier\_message", "unknown" or a list them - separated by plus sign ("+") or dot (".") - drops the call when detect voice mail box, human answer, answering machine messages, operator messages, or unknown answer pattern - respectively. Available in digital signals (E1 links and devices KGSM). Additionally, the information service is reported to the user in the variable **KCallAnswerInfo**, being necessary to patch the **Asterisk** for this variable can be accessed;
- **input\_volume**: Sets the volume of inbound link (ranges from -10 to +10);
- **no\_parent\_cid**: Request to channel not to import the callerid from the requestor of the call (when exists) in case of the actual channel callerid is empty;
- **orig**: Sets the originator number, **without changing the variable  $\${CALLERID(num)}$** . Beware that if Asterisk has already set the variable  **$\${CALLERID(num)}$** , which is the default behavior, the channel automatically uses this value as the originator number, without having to pass any additional options (like this one).  
On **KGSM** devices, if set to **restricted**, the originator number will be omitted (if the operator allows this option to be used). Example:

```
Dial(Khomp/b0/99887766/orig=restricted)
```

**WARNING:** In KGSM channel devices, the only valid value for this option is **restricted**.

- **output\_volume**: Sets the output volume of the link (ranges from -10 to +10);
- **parent**: Set the name of the channel which requested this outgoing call, allowing status variables to be adjusted on it (not needed for Asterisk version 1.8 or higher);

This resource is available since version: **3.0**.

- **pre**: When set to a string of DTMF digits, uses these to pre-allocate an output channel in an analog PABX, dial the desired number of B below. Only available for signaling analog (FXO);
- **pre\_answer**: When set (need no value), answers the channel before the connection is completed - allowing, for instance, DTMF tones to sent (useful for use in a **DISA**);
- **ring\_cadence**: When set to a cadence name (listed in the **[cadences]** section), uses this for ringing FXS channels;
- **ring**: When set to two numbers separated by a dot ("."), defines the cadences to be used while ringing a FXS channel - the first time is the ringing time, and the second one, the silence time;
- **ring\_ext**: When set to two numbers separated by a dot ("."), defines the extended cadences to be used while ringing



a FXS channel, executed after the **ring** specification - the first time is the ringing time, and the second one, the silence time;

- **usr\_xfer**: Defines a group of DTMF digits to initiate a transfer between PBXes (using QSig or FXO FLASH, for instance);
- **uii**: When adjusted for a number and a string of text, separated by hash ("#"), sends a "UserToUser" to the other end before making the call - the first value will be the descriptor and the second one will be the message as the text (available only in ISDN signaling);

## List of variables

Here's a list of available variables:

Nome	Tipo	Descrição
KDropCollectCall	W/O	When set before ringing or answer an incoming call, enables ("yes") or disables ("no", default) the drop of collect calls based on signaling received from the central public, double answer, audio tone recognition (can be defined globally) ( This resource is available since version: <b>2.2.</b> );
KR2GotCategory	R/O	Adjusted by the channel when an incoming call is received, with the category number of the caller (only available on R2 signaling). We recommend using a 2 seconds delay before call this variable (Wait(2)). For Asterisk versions before 1.8, it requires a patch in Asterisk, or to adjust the "parent" Dial option accordingly on the outgoing call. Available only for R2/MFC signaling.;
KR2StrCategory	R/O	Adjusted by the channel when an incoming call is received, with the category name of the A number. (only available on R2 signaling). We recommend using a 2 seconds delay before call this variable (Wait(2)). For Asterisk versions before 1.8, it requires a patch in Asterisk, or to adjust the "parent" Dial option accordingly on the outgoing call. Available only for R2/MFC signaling.;
KR2GotCondition *	R/O	Adjusted by the channel, available after returning from a call made by Asterisk, has the numeric value representing the condition of the remote end, received when making the call (for Asterisk versions <b>before 1.8</b> , it <b>requires</b> a patch in Asterisk, or to adjust the " <b>parent</b> " Dial option accordingly on the outgoing call) (available only for R2 signaling);
KR2StrCondition*	R/O	Adjusted by the channel, available after returning from a call made by Asterisk, has the name value representing the condition of the remote end, received when making the call (for Asterisk versions <b>before 1.8</b> , it <b>requires</b> a patch in Asterisk, or to adjust the " <b>parent</b> " Dial option accordingly on the outgoing call) (available only for R2 signaling);
KISDNGotCause *	R/O	Adjusted by the channel, available after returning from a call made by Asterisk, has the numeric value for the ISDN "cause" received when making the call (for Asterisk versions <b>before 1.8</b> , it <b>requires</b> a patch in Asterisk, or to adjust the " <b>parent</b> " Dial option accordingly on the outgoing call) (available only for ISDN signaling);
KISDNStrCause*	R/O	Adjusted by the channel, available after returning from a call made by Asterisk, has the name value for the ISDN "cause" received when making the call (for Asterisk versions <b>before 1.8</b> , it <b>requires</b> a patch in Asterisk, or to adjust the " <b>parent</b> " Dial option accordingly on the outgoing call) (available only for ISDN signaling);
KCallAnswerInfo *	R/O	Adjusted by the channel, available after returning from a call made by Asterisk, contains the service information identified by the call analyser (for Asterisk versions <b>before 1.8</b> , it <b>requires</b> a patch in Asterisk, or to adjust the " <b>parent</b> " Dial option accordingly on the outgoing call) (available only for digital signaling - E1 and GSM);
KR2SendCondition (deprecated)	W/O	When set to a numeric value, before the ringing an incoming call, adjusts the condition for this channel to this value (available only on R2 signaling);
KISDNSendCause (deprecated)	W/O	When set to a numeric value, before the ringing an incoming call, adjusts the cause for this channel to this value. (available only for ISDN signaling); This resource is available since version: <b>3.0.</b>
KR2Condition	W/O	When set to a numeric value, before the ringing an incoming call, adjusts the B condition for this channel to this value. (available only on R2 signaling); This resource is available since version: <b>3.0.</b>
KISDNCause	W/O	When set to a numeric value, before the ringing an incoming call, adjusts the cause for this channel to this value. (available only for ISDN signaling); This resource is available since version: <b>3.0.</b>
KR2Category	R/W	Adjusted automatically on incoming calls, contains the originator category. When set to a numeric value, before making a call by Asterisk, adjusts the A category for this channel. The variable is automatically passed between two R2 signaling channels, because is defined as inheritable. (available only on R2 signaling);

		<div>This resource is available since version: <b>3.0</b>.</div>
<b>KISDNOrigTypeOfNumber</b>	R/W	Adjusted automatically on incoming calls, contains the type of the originator number. When set to a numeric value, before making a call by Asterisk, adjusts the type of the originator number for this channel. The variable is automatically passed between two ISDN signaling channels, because is defined as inheritable. (available only for ISDN signaling); <div>This resource is available since version: <b>3.0</b>.</div>
<b>KISDNDestTypeOfNumber</b>	R/W	Adjusted automatically on incoming calls, contains the type of the destination number. When set to a numeric value, before making a call by Asterisk, adjusts the type of the destination number for this channel. The variable is automatically passed between two ISDN signaling channels, because is defined as inheritable. (available only for ISDN signaling); <div>This resource is available since version: <b>3.0</b>.</div>
<b>KISDNOrigNumberingPlan</b>	R/W	Adjusted automatically on incoming calls, contains the originator's numbering plan. When set to a numeric value, before making a call by Asterisk, adjusts the originator's numbering plan for this channel. The variable is automatically passed between two ISDN signaling channels, because is defined as inheritable. (available only for ISDN signaling); <div>This resource is available since version: <b>3.0</b>.</div>
<b>KISDNDestNumberingPlan</b>	R/W	Adjusted automatically on incoming calls, contains the destination's numbering plan. When set to a numeric value, before making a call by Asterisk, adjusts the destination's numbering plan for this channel. The variable is automatically passed between two ISDN signaling channels, because is defined as inheritable. (available only for ISDN signaling); <div>This resource is available since version: <b>3.0</b>.</div>
<b>KISDNOrigPresentation</b>	R/W	Adjusted automatically on incoming calls, contains the presentation type of the originator number. When set to a numeric value, before making a call by Asterisk, adjusts the presentation type of the originator number for this channel. The variable is automatically passed between two ISDN signaling channels, because is defined as inheritable. (available only for ISDN signaling); <div>This resource is available since version: <b>3.0</b>.</div>
<b>KSmsDelivered</b>	R/O	Adjusted by the channel when sending a SMS message with the application <b>KSendSMS</b> , saying whether the message was delivered successfully ("yes") or not ("no");
<b>KSmsErrorCode</b>	R/O	Adjusted by the channel when sending a SMS message with the application <b>KSendSMS</b> , containing the error code that happened when sending the message. <u>Until channel 2.4.1 this variable had a string describing the error name, since channel 3.0, it contains the error code number;</u>
<b>KSmsErrorName</b>	R/O	Adjusted by the channel when sending a SMS message with the application <b>KSendSMS</b> , contains the name of the error or "None" if there has been no error. ( <div>This resource is available since version: <b>3.0</b>.</div> );
<b>KSmsType</b>	R/O	Adjusted for the input channel in the context of SMS messages, defines the type of message received (can contain the values "message", "confirm" or "broadcast". ( <div>This resource is available since version: <b>3.0</b>.</div> );
<b>KSmsFrom</b>	R/O	Adjusted for the input channel in the context of SMS messages, sets the number of origin of the received message (available on types "message" and "confirm");
<b>KSmsDate</b>	R/O	Adjusted for the input channel in the context of SMS messages, sets the date of sending the message (available on types "message" and "confirm");
<b>KSmsSize</b>	R/O	Adjusted for the input channel in the context of SMS messages, contains the size (in

<b>KSmsSize</b>	R/O	bytes) of the received message (available on types "message" and "broadcast");
<b>KSmsMode</b>	R/O	Adjusted for the input channel in the context of SMS messages, contains the encoding type of the received message (available on types "message" and "broadcast");
<b>KSmsBody</b>	R/O	Adjusted for the input channel in the context of SMS messages, contains the text of the received message (available in types "message" and "broadcast");
<b>KSmsDelivery</b>	R/O	Adjusted for the input channel in the context of SMS messages, containing the date of delivery of the message sent earlier (available in type "confirm") ( This resource is available since version: <b>3.0.</b> );
<b>KSmsStatus</b>	R/O	Adjusted for the input channel in the context of SMS messages, contains the status of the message sent earlier (available on the type "confirm") ( This resource is available since version: <b>3.0.</b> );
<b>KSmsSerial</b>	R/O	Adjusted for the input channel in the context of SMS messages, contains the serial number of the received message (available on the type "broadcast") ( This resource is available since version: <b>3.0.</b> );
<b>KSmsPage</b>	R/O	Adjusted for the input channel in the context of SMS messages, contains the page number of the received message (available on the type broadcast") ( This resource is available since version: <b>3.0.</b> );
<b>KSmsPages</b>	R/O	Adjusted for the input channel in the context of SMS messages, contains the total number of pages to be received (available in type "broadcast") ( This resource is available since version: <b>3.0.</b> );
<b>KUssdDelivered</b>	R/O	Adjusted by the channel when sending a USSD message with the application <b>KSendUSSD</b> , saying whether the message was delivered successfully ("yes") or not ("no") ( This resource is available since version: <b>4.2.2.</b> );
<b>KUssdErrorCode</b>	R/O	Adjusted by the channel when sending a USSD message with the application <b>KSendUSSD</b> , containing the error code that happened when sending the message ( This resource is available since version: <b>4.2.2.</b> );
<b>KUssdErrorName</b>	R/O	Adjusted by the channel when sending a USSD message with the application <b>KSendUSSD</b> , contains the name of the error or "None" if there has been no error ( This resource is available since version: <b>4.2.2.</b> );
<b>KUssdMessage</b>	R/O	Adjusted for the input channel in the context of USSD messages, contains the text of the received message ( This resource is available since version: <b>4.2.2.</b> );
<b>KTDDStringReceived</b>	R/O	Adjusted for the input channel in the context of TDD messages, contains the text of the received message by KReceiveTDD application ( This resource is available since version: <b>3.1.</b> ) (This feature is <b>deprecated</b> since version <b>4.0.</b> ).
		Adjusted automatically on incoming User-to-User Information message, defines the command to be used to send the message <b>UUI</b> : extended command (up to 256

<b>KUserInfoExtended</b>	R/W	characters) or standard (up to 32 characters). When set to a boolean value, before making a call by Asterisk, defines the command to be used to send the message <b>UUI</b> . The variable is automatically passed between two ISDN signaling channels, because is defined as inheritable. (available only for ISDN signaling);
<b>KUserInfoDescriptor</b>	R/W	Adjusted on incoming User-to-User Information message, contains the protocol descriptor used in the message. When set to a numeric value, before making a call by Asterisk, adjusts the protocol descriptor of the <b>UUI</b> message. The variable is automatically passed between two ISDN signaling channels, because is defined as inheritable. (available only for ISDN signaling);
<b>KUserInfoData</b>	R/W	Adjusted on incoming User-to-User Information message, contains the message data in format base 16 (characters represented in hexadecimal). When set to a message in format base 16, before making a call by Asterisk, adjusts the data of the <b>UUI</b> message. The variable is automatically passed between two ISDN signaling channels, because is defined as inheritable. (available only for ISDN signaling);
<b>KUserTransferStatus</b>	R/O	Returns the state for the transfer command ( <b>KUserTransfer</b> ) in <b>FXO</b> signaling, when the <b>W</b> option is used. Can be set to: <ul style="list-style-type: none"> <li>▪ <b>SUCCESS</b>: Answer was detected.</li> <li>▪ <b>TIMEOUT</b>: Answer was not detected at the specified time.</li> <li>▪ <b>HANGUP</b>: The call was disconnected while the answer was expected.</li> </ul>
<b>KFaxSent</b>	R/O	Adjusted by the channel when sending FAX with <b>KSendFax</b> application, and determines whether the fax was successfully sent ("yes") or not ("No"); (This feature is <b>deprecated</b> since version <b>4.0</b> ).
<b>KFaxReceived</b>	R/O	Adjusted by channel when receiving FAX with <b>KReceiveFax</b> application, and determines whether the fax was successfully received ("yes") or not ("no"); (This feature is <b>deprecated</b> since version <b>4.0</b> ).
<b>KFaxResult</b>	R/O	Adjusted by the channel when sending or receiving FAX with the application <b>KSendFax</b> or <b>KReceiveFax</b> (respectively), and defines the result of execution; (This feature is <b>deprecated</b> since version <b>4.0</b> ).
<b>KOutgoingChannel</b>	R/O	Adjusted by the channel on outgoing calls, contains the device and channel will be used to make this call; ( <div style="border: 1px solid black; padding: 2px; display: inline-block;">This resource is available since version: <b>3.0</b>.</div> ).

## Description of variables

Below, follows an explanation on how to use variables of Khomp channel in the dialplan, to communicate and/or to receive information:

### KDropCollectCall

When activated, causes the channel to drop Khomp collect calls through dual service (available for signaling 'R2 Digital' and FXO), through information available on the ISDN protocol and R2/MFC, or by detecting the audio call collect (available for any digital signage for E1, and GSM signaling).

This variable is useful to filter collect calls to certain extensions, and **must** be set before making any type of answer - applications such as **Playback** and **Dial** should always be executed after setting this variable, for example.

For better functionality, is also recommended that no call status (ringback) is sent before this variable is set, so applications such as **Progress** or **Ringin** also should be performed only after the correct setting of this variable.

This variable can be set locally and globally, both to **yes** or **no**. The adjustment of global variable **toyes** will drop all the collect calls, unless the particular call is set to **no** - this allows the creation of a global filter of collect calls, with few exceptions.

Enabling the variable in context **[default]**:

```
[default]
...
exten => _X.,1,Set (KDropCollectCall = yes)
...
```

Enabling the variable in the global context:

```
[globals]
...
KDropCollectCall = yes
...
```

## KR2Condition

When you receive a call, can be set before sending ringback by Asterisk (ie, before the run Asterisk applications **Answer**, **Ringing**, or **Dial**). When used in signaling R2/MFC, this variable sets the condition for B to the numeric value desired.

Example:

```
;; Condition "NUMBER CHANGED" warns the caller that the number of B has changed.
;;
exten => _X.,1,KR2Condition (3)
```

## KR2GotCategory

When you receive a call, is set by the channel with the category received from the number that originated the call. It is set in signaling R2/MFC, and can be found anywhere in the dialplan.

Example:

```
exten => _X.,1,NoOp(${KR2GotCategory})
```

## KR2GotCondition

Set by the channel and available after returning from a call made by Asterisk, holding the number which represents the condition of B received when making the call. We recommend using a 2 seconds delay before call this variable (Wait(2)). For Asterisk versions **before 1.8**, it **requires** a patch in Asterisk, or to adjust the **"parent"** Dial option accordingly on the outgoing call. Available only for R2/MFC signaling.

Example:

```
exten => _X.,1,NoOp(${KR2GotCondition})
```

## KUserInfoDescriptor

Variable adjusted by the channel in the context of entry, from information received by the ISDN network functionality through user-to-User Information. Contains the descriptor number of the protocol used by the other end, and usually contains the value '0 ', but this is dependent on application.

For further information, consult the specification ITU-T Q931 (more precisely, the specification table 4-26).

Example (working with the descriptor number of the protocol):

```
exten => _X.,1,Verbose(${KUserInfoDescriptor})
```

## KUserInfoData

Variable adjusted by the channel in the context of entry, from information received by the ISDN network functionality through user-to-User Information. Contains the actual data, which were received in the form of a string of text.

More information about this feature, see the specification ITU-T Q931.

Example (working with the data received):

```
exten => _X.,1,NoOp(${KUserInfoData})
```

It is important to note that the variables are sensitive to the capitalization of letters.

## KCallAnswerInfo

Set by the channel in outbound connections, representing the type of answer performed by the other end. For Asterisk versions **before 1.8**, it **requires** a patch in Asterisk, or to adjust the "**parent**" Dial option accordingly on the outgoing call.

May contain the following values:

- \* "MessageBox" (\*): detected mailbox of a cell phone;
- \* "CarrierMessage": message sent before the service provider;
- \* "AnsweringMachine" (\*\*): answering answering machine;
- \* "HumanAnswer" (\*\*): human service;
- \* "Fax": reported when a fax tone is detected.
- \* "Unknown": unknown type of care;

(\*) This type of service is detected by signals at certain frequencies that are sent before the call comes into a mailbox, and vary by operator. The algorithm captures most of the mailboxes, but can fail if there is not a clear signal, or if it is not within the standards most commonly used;

(\*\*) The difference between these two types of care depends on the specific configuration using the program **k3lanswerinfoconfig**, with the detection only based on heuristics and **never** with an accuracy of 100%.

## KOutgoingChannel

Variable adjusted by the channel on outgoing calls, contains which channel was allocated to the connection in the BXCXY format, where X represents the device number and Y the channel number.

**NOTE:** This variable can only be reported to the originator channel if the **parent** parameter is set on the **Dial**.

```
exten => 100,1,Dial(khomp/b010/${EXTEN}/parent=${CHANNEL},30,ht)
```

```
exten => h,1,NoOp(Allocated channel on outgoing call: ${KOutgoingChannel} )
```

## KTDDStringReceived

Variable adjusted with the received message by KReceiveTDD application. (

This resource is available since version: **3.1**.

) (This feature is **deprecated** since version **4.0**.)

```
exten => 100,1,KReceiveTDD()  
exten => 100,2,NoOp(${KTDDStringReceived})
```

## CHANNEL() function

The CHANNEL() function allows to obtain both values as set in the channel in question. The channel's Khomp supports the following values (

This resource is available since version: **3.0**.

):



Name	Type	Description
collectcall	R/O	Returns "yes" if this is an incoming collect call, or "no" if not (or if the signaling does not provide this information).
gsmantenna	R/O	Returns the signal level of the device, from 0% to 100%, or "unknown" if unknown.
gsmerrorrate	R/O	Returns the index of errors in signal transmission from 0% to 7%, or "unknown" if unknown.
gsmoperatorname	R/O	Returns the name of the operator where the SIM card is currently registered.
gsmregistrystatus	R/O	Returns registration status of the SIM card, which can be "notregistered", "registered", "searching", "denied", "roaming", "initializing" or "unknown".
gsmsimcard	R/W	Returns the number of the SIM card currently in use, or adjusts the SIM card to the integer passed as value.
inputVolume	R/W	Read or adjusts the volume of the current channel (from -10 to +10).
outputVolume	R/W	Read or adjusts the volume of the current channel (from -10 to +10).

Example: Reading values

```
exten => _X.,1,Set(var_collectcall=${CHANNEL(collectcall)})
```

Example: Setting values (Read/Write or R/W variables)

```
exten => _X.,1,Set(CHANNEL(inputvolume)=+5)
```

## Console commands

List of available commands in the console for the Asterisk channel of Khomp:

### khomp channels disconnect

Disconnect one or more channels currently connected. This command sends a message directly to the physical channel of the device in question, requesting a disconnection. Use with caution;

### khomp calls show

Shows states for calls, may also listing specific channels or devices (

This resource is available since version: **3.0**.

);

The identifier (**hw id**) consists of the device number, number of physical channel (on device), logical channel number and call number. GSM channels have 5 logical channels and one conference channel. The other signaling have only one logical channel. All logical channel have space for two calls.

The **chan** and **call** fields consist of the logical channel state and call state. The **orig\_addr** and **dest\_addr** fields inform the originator and destination number - respectively.

The **info** field shows extra information about the channel, and the options are:

- **M**: the GSM call is in a **Multiparty** conference; (This feature is **deprecated** since version **4.0**.)
- **O**: the call has one Asterisk channel associated.

### khomp channels show

Shows the status of the channels Khomp and may also list specific adapter;

### khomp channels statistics

Shows the statistics of channel connections, or statistics for a particular channel;



## khomp channels statistics clear

Clears the channels statistics, or specific link or channel statistics;

## khomp channels unblock

Unlock blocked channels for input or output. Only available in digital signage via E1 ;

## khomp dump config

Dump general and local configuration options (and values) to the screen (

This resource is available since version: **3.0**.

);

## khomp dump branches

Shows configured branches on FXS devices (

This resource is available since version: **3.0**.

).

## khomp kommuter

Enables or disables kommuters connected via USB on this machine. Only accessible when the configuration "kommuter-activation" is set to "manual" (

This resource is available since version: **2.4.1**.

);

## khomp kommuter count

Gets the amount of kommuters connected via USB on this machine (

This resource is available since version: **2.4.1**.

).

## khomp links errors

Shows status of error counters for all channels, or for a device/link only;

## khomp links reset

Sends a reset command for a specific **E1** of a particular device;

## khomp links show

Display states of E1 links available;

## khomp log console

Sets options in the console logs;

## khomp log disk

Sets logging options to disk;

**khomp log console** and **khomp log disk** have auxiliary options **No**, which reverses the selection of messages, and **just**, which generalizes the choice.

Examples:

```
khomp log disk just commands events
; Enables *only* logging to the disk of commands and events;
```

```
khomp log disk no commands
; Disable logging to the disk of commands sent to API;
```

```
khomp log disk warnings
; Enables also logging to the disk of warnings from channel;
```

- More information on options for the **log** command, type: **help khomp log disk** or **help khomp log console**.

## khomp log rotate

Rotate log files from the channel;

## khomp log update

Update log configuration from K3L (klog.cfg) (

This resource is available since version: **4.0**.

).

## khomp log status

Shows log messages currently being written to disk and displayed on the console;

## khomp log trace isdn

Enable Debugging ISDN signaling (ISDN) (This feature is **deprecated** since version **4.0**).

## khomp log trace k3l

Enables debugging low-level API K3L (This feature is **deprecated** since version **4.0**).

## khomp log trace r2

Enables debugging low-level signaling R2/MFC (This feature is **deprecated** since version **4.0**).

## khomp modem reset

Reset the modem of the GSM channel passed (

This resource is available since version: **3.0**.

);

## khomp modem sim card

Shows the number of the active SIM card on selected GSM channel (

This resource is available since version: **3.0**.

);

## khomp record

Enable audio recording through the Khomp channel (This feature is **deprecated** since version **4.0**);

## khomp revision

Shows version number and revision of the channel;

## khomp select sim

Select the SIM card, available on the GSM devices (

This resource is available since version: **2.4.1**.

);

## khomp modem imei

Shows the modem IMEI on GSM devices (

This resource is available since version: **3.0**.

).

## khomp modem imsi

Shows the SIM card IMSI on GSM devices (

This resource is available since version: **4.2.3**.

).

## khomp modem iccid

Shows the SIM card ICCID on GSM devices (

This resource is available since version: **4.2.3**.

).

## Khomp send command

Sends command API K3L directly to the device (only for debugging, may compromise the stability of the system if used improperly);

## Khomp send raw command

Sends a command directly to the DSP board (only for debugging, may compromise the stability of the system if used improperly) (This feature is **deprecated** since version **4.0**.);

## khomp get

Shows various options of the channel Khomp, explained in greater detail in the description of use of the command ("help khomp get");

## khomp set

Sets various options of the channel Khomp, explained in greater detail in the description of use of the command ("help khomp set");

## khomp sms

Sends an SMS message for a given number, using KGSM channels;

Usage: khomp sms <device> <destination> <message>

Fields have the following meaning:

- **device**: Device to use (same string used in Dial for channel allocation).
- **destination**: Number where to send the message.
  - **option**: Put the **!** character at the end of the destination to request a confirmation of transmission.
  - **option**: Put the **a** character at the end of the destination to send the SMS as an alert message.
- **message**: Message to send.

Example:

```
khomp sms b0 99887766 I will send an SMS to the world!
khomp sms b0c10 99887766! Message requesting confirmation.
```

## khomp summary

Prints a summary of system devices and their features;

## khomp ussd

Sends a USSD message, using KGSM channels;

Usage: khomp ussd <device> <message>

- **device**: Device to use (same string used in Dial for channel allocation).
- **message**: Message to send.

Example:

```
khomp ussd b0 *123#
```

For more information and usage examples, add **help** before the commands. For instance: **help khomp links errors**.

# Additional features

This chapter discusses additional features of the channel, related to the special features present in certain signs.

## Applications (applications) and channels

The channel's Khomp, and to record a type of communication channel "Khomp" also records the following items:

### "KUserTransfer" application

Performs the transfer process from the current channel number for the extension using the QSig signaling protocol (Single Step Call Transfer) for devices configured with E1 ISDN signaling (ISDN), or the FLASH command for FXO channels.

The syntax follows:

```
KUserTransfer(number[,options])
```

Example:

```
exten => 1234,1,KUserTransfer(2345)
```

The fields have the following meanings:

- **number**: Number where the link should be transferred.
  - This parameter may be "**none**" or omitted in **FXO** devices, where it sends only the transfer command (FLASH) to line.

This resource is available since version: **3.0**.

- **options**: Sets the transfer options to be used, which are:
  - **k**: Continues execution of the *dialplan* after sending the transfer command (available only for **FXO** signaling).

This resource is available since version: **3.0**.

- - **t**: Specifies the protocol **facility** being used to transfer, and may contain the same values ??as the **qsig-transfer-facility** configuration option (available only for **ISDN** signaling).

This resource is available since version: **3.0**.

- **W**: Wait until the channel is disconnected (available only for **ISDN** signaling).
- **W(n)**: Wait for the answer or disconnection for up **n** seconds (available only for **FXO** signaling).  
**IMPORTANT**: for **FXO** channels, the waiting uses voice detection to validate the answer, which in some cases may require adjustments in the voice detection parameters of the device. The result of the execution can be obtained by the **KUserTransferStatus** variable.

This resource is available since version: **3.0**.

## "KRecord" application

(This feature is **deprecated** since version **4.0**.);

Application to record audio through the boards Khomp. This application was created to use the recording feature in this Khomp boards, which can be used to mix the audio hardware and send it directly to disk. The recording is done **only** in the **A-Law** format with **WAV** headers. If the input channel is not a channel Khomp and the option **f** has not been informed, the KRecord invokes the application **MixMonitor** to make the recording.

The syntax of the application is as follows:

```
KRecord([filename [| options [| other-options]])
```

The fields have the following meanings:

- **file**: Name of recording and, if omitted, will use a standard name format.
- **options**: Sets the recording options to be used, which are:
  - **b**: Records only when channel is connected (similar to option **"b"** of **MixMonitor**);
  - **f**: Force the application **KRecord** to run, even if the input channel is not of type **Khomp**. In this case, the application sets the variable current channel so that when the output channel has audio available, the recording starts. **WARNING: This option can only be used when the output channel is of type Khomp. If this prerequisite is not met, the recording will not be done**;
  - **R(<command> <parâmetro1> <parâmetro2> ...)**: Executes <command> the end of the recording, as an operating system command, passing as parameters <parâmetro1>, followed by <parameter2> ... (if any);
  - **s**: Record in stereo, with one track used for audio sent to the board, and other used for audio read from the board;
  - **w**: Stop dialplan execution, waiting until the end of the connection to continue (useful for passive recording boards).
- **other-options**: Sets the recording options to be passed to the application **Monitor** or **MixMonitor** where this is called the **KRecord** (the application choice depends on the option **s**: if its set, **Monitor** will be chosen; if its not, **MixMonitor** will be used instead).

Examples:

```
exten => 1234,1, KRecord (test.wav)
exten => 1234,1, KRecord(test.wav|f)
```

## "KStopRecord" application

(This feature is **deprecated** since version **4.0**.);

Application to stop the audio recording previously started with **KRecord**. The syntax of the application is as follows:

```
KStopRecord()
```

This application does not receive parameters. Example of use:

```
exten => 1234,1,KStopRecord()
```

## Application "KSendSMS"

This application has the function of sending SMS messages through the KGSM Khomp devices using its modules and SIM cards. The syntax of the application is as follows:

```
KSendSMS (resource | destination [/options][|message])
```

Each field can be summarized in:

- **resource**: Follows a similar syntax to the channel allocation of the Dial application, and defines which modem to use;
- **destination**: Number where to send the message, may be preceded or succeeded by! to request a confirmation of transmission;
- **options**: Options to set parameters or send specials SMS:
  - **a**: Sends SMS as alert message. Also called SMS Flash or SMS of class 0;
  - **c**: Requests for the operator a confirmation message sending SMS;
  - **e(<encoding>)**: Sets the encoding of the SMS, where<encoding> can be:
    - **7**: Sends the message in GSM 7 bits encoding - default encoding;
    - **binary**: Sends the message in binary encoding;
    - **16**: Sends the message in GSM 16 bits encoding. **Note: This encoding is only available for sending WAP type message**;
  - **w(<URL>)**: Requests to send a SMS message of the WAP PUSH type with URL<URL>. This message can be sent in 8 or 16 bit encoding. The types of WAP message are:
    - **Service Indication (SI)**: Sends a link to the URL and a text in the message;
    - **Service Load (SL)**: Sends only the link to the URL. The **message** field must be empty;
  - **p(<port>)**: Sets the URL port number sent in WAP message. The default port is 2948;
- **message**: Text (without quotes) to be sent to**destination**. If the **message** parameter is not provided, the SMS is sent without a message;

After sending the message, the variables **KSmsDelivered** and **KSmsErrorCode** will contain the result of the post. For more information about these, please consult the section on variables used in the channel.

Examples of use of this application are as follows:

- Sends "Test message." for "99887766" using the channel "1" (second modem) of the device "0":

```
exten => [...],1,NoOp (Sending SMS ...)  
exten => [...],n,KSendSMS (b0c1 | 99,887,766 | Test message.)
```

- Sends "Test message." for "99887766" using the first free channel of the device "0", and checks the return:

```
exten => [...],1,NoOp(Sending SMS ...)  
exten => [...],n,KSendSMS(b0|99887766|Message test.)  
exten => [...],n,NoOp(sent? ${KSmsDelivered})  
exten => [...],n,NoOp(Code: ${KSmsErrorCode})  
exten => [...],n,NoOp(Desc: ${KSmsErrorName}); (
```

This resource is available since version: **3.0**.

- Sends "Test message." for "99887766" using the first free channel of the device "0", or the first free channel of the device "1" (if no free channel at the first device):

```
exten => [...],1,NoOp(Sending SMS ...)  
exten => [...],n,KSendSMS(b0+b1|99887766|Test message.)
```

- Sends "Test message." for "99887766" using the first free channel of the device "0", requesting confirmation:

```
exten => [...],1,NoOp(Sending SMS ...)  
exten => [...],n,KSendSMS(b0|99887766/c|Test message.)
```

- Sends "Test message." to "99887766" as alert message using the first free channel of the device "0", requesting confirmation:

```
exten => [...],1,NoOp(Sending SMS...)  
exten => [...],n,KSendSMS(b0|99887766/ac|Test message.)
```

- Sends "Visit this site:" text and "www.khomp.com.br" link to "99887766" as WAP (SI) message using the first free channel of the device "0", requesting confirmation:

```
exten => [...],1,NoOp(Sending WAP: text + link...)  
exten => [...],n,KSendSMS(b0|99887766/e(binary)cw(www.khomp.com.br)|Visit this site:)
```

- Sends "www.khomp.com.br" link and "1234" port to "99887766" WAP (SL) message using the first free channel of the device "0",

```
exten => [...],1,NoOp(Sending WAP: link...)  
exten => [...],n,KSendSMS(b0|99887766/e(binary)w(www.khomp.com.br)p(1234))
```

## Application "KSendUSSD"

This application has the function of sending USSD messages through the KGSM Khomp devices using its modules and SIM cards. The syntax of the application is as follows:

```
KSendUSSD (resource | message)
```

Each field can be summarized in:

- **resource**: Follows a similar syntax to the channel allocation of the Dial application, and defines which modem to use;
- **message**: Text (without quotes) to be sent;

After sending the message, the variables **KUssdDelivered**, **KUssdErrorCode**, and **KUssdErrorName** will contain the send result. For more information about these, please consult the section on channel variables.

Example of use:

- Sends "\*123#" using the first free channel of device "0", and checks the return:

```
exten => [...],1,NoOp(Sending USSD ...)  
exten => [...],n,KSendUSSD(b0|*123#)  
exten => [...],n,NoOp(sent? ${KUssdDelivered})  
exten => [...],n,NoOp(Code: ${KUssdErrorCode})  
exten => [...],n,NoOp(Desc: ${KUssdErrorName})
```

## Application "KEchoCanceller"

This application has the function to enable or disable the echo canceller channel.

Syntax updated application (

This resource is available since version: **2.4.1**.

)

```
KEchoCanceller(action[,options])
```

Previous syntax:

```
KEchoCanceller (action)
```

Where:

- **actions:** It is **on** to enable the echo canceller, and **off** to disable;
- **options:** Sets the options of *application*, which are:
  - **N:** Perform the action only in the *current* channel, not all call.

Example usage of this application:

```
exten => [...], n KEchoCanceller (off)
```

## Application "KAutoGainControl"

This application has the function to enable or disable the automatic gain control in the channel.

Syntax updated (

This resource is available since version: **2.4.1**.

):

```
KAutoGainControl (action[,options])
```

Previous syntax:

```
KAutoGainControl (action)
```

Where:

- **actions:** It is **on** to enable the automatic gain control, and **off** to disable;
- **options:** Sets the options of *application*, which are:
  - **N:** Perform the action only in the *current* channel, not all call.

Example usage of this application:

```
exten => [...], n, KAutoGainControl (on)
```

## Application "KDTMFSuppression"

This application has the function to enable or disable the suppression of DTMF channel. The syntax of the application is as follows:

Syntax updated (

This resource is available since version: **2.4.1**.

):

```
KDTMFSuppression (action[,options])
```

Previous syntax:

```
KDTMFSupression (action)
```

Where:

- **actions:** It is **on** to enable DTMF suppression, and **off** to disable;
- **options:** Sets the options of *application*, which are:



- **N**: Perform the action only in the *current* channel, not all call.

It is important to note that when suppression of DTMF tones is disabled, the tones are passed inband and **will not be reported to Asterisk**. Thus, Asterisk will not recognize the DTMF tones, and will be unable to run applications such as IVR and execute features recognition.

Example usage of this application:

```
exten => [...],n,KDTMFSuppression (off)
```

## Application "KSetVolume"

This application has the function to adjust the volume of incoming and outgoing channels Khomp, and its syntax as follows:

```
KSetVolume (<volume>)
KSetVolume (<output-volume>|<input-volume>)
```

Where the fields have the following meanings:

- **volume**: Sets the volume of input and output (-10 to +10);
- **output-volume**: Sets the output volume (-10 to +10, "none" for no change);
- **input-volume**: Sets the input level (-10 to +10, "none" for no change).

## Application "KAdjustForFax"

This application has the function of setting a channel for receiving Khomp signal FAX/modem, optimizing the communication channel for data traffic. Syntax:

```
KAdjustForFax()
```

This application does not receive parameters. Example of use:

```
exten => 1234,1,KAdjustForFax ()
```

## Application "KSendFax"

(This feature is **deprecated** since version **4.0.**);

This application has the function to send faxes using digital channels or FXO connections Khomp in pre-established, and its syntax as follows:

```
KSendFax (<file> [:<arquivo2> [:...]][:< faxid>])
```

**This application requires a license purchased separately to be used in digital (non-FXO) channels.** Fields have the following meanings:

- **file**: Files to be sent to the fax should be encapsulated in TIFF format and have a resolution of 98, 196 or 392 dpi;
- **faxid**: the fax number. If not specified, the value is obtained by the id of the link, and if this also is not valid, the fax number will be set as default in K3L.

Example usage of this application:

```
exten => [...],1,KSendFax (/tmp/fax.tif:/home/root/fax2.tif,1234)
```

## Application "KReceiveFax"

(This feature is **deprecated** since version **4.0.**);

This application has the function of receiving digital channels or fax using the FXO Khomp, and its syntax as follows:

```
KReceiveFax (<file> [| <faxid>])
```

**This application requires a license purchased separately to be used in digital (non-FXO) channels.** Fields have the following meanings:

- **file**: Name that file will be assigned to incoming fax.
- **faxid**: the fax number. If not specified, the value is obtained by the id of the link, and if this also is not valid, the fax number will be set as default in K3L.

Example usage of this application:

```
exten => [...],1,Answer()  
exten => [...],n,KReceiveFax(/tmp/fax.tif)
```

## Application "KSelectSimCard"

This application has the function to select SIM card in khomp devices, and its syntax as follows:

```
exten => [...],n,KSelectSimCard([<device>|<channel>|]<sim_card>)
```

Where the fields have the following meanings:

- **device**: Defines which device to send the command.
- **channel**: Defines which channel in the device to send the command.
- **sim\_card**: The SIM card number to be selected.

Example:

```
exten => [...],1,Answer()  
exten => [...],n,KSelectSimCard(0|2|1)
```

## Aplicação "KSendTDD"

(This feature is **deprecated** since version **4.0.**)

This application has the function to send text using **TDD** feature (

This resource is available since version: **3.1.**

), and its syntax as follows:

```
KSendTDD (<text>)
```

Where:

- **text**: Maximum of 255 characters

Example:

```
exten => [...], n, KSendTDD(Hello world!)
```

## Aplicação "KReceiveTDD"

(This feature is **deprecated** since version **4.0**.)

This application has the function to receive text using **TDD** feature (

This resource is available since version: **3.1**.

), and its syntax as follows:

```
KReceiveTDD([timeout])
```

Where:

- **timeout**: Ajusta o tempo limite de espera dos caracteres, *default* igual a 5 segundos.

## Channel "Khomp\_SMS"

This communication channel is used to receive SMS and create incoming links in Asterisk for each message received. This channel does not have any treatment or audio processing, and is called with five variables set:

- **KSmsFrom**, containing the number of source who sent the message;
- **KSmsDate**, which sets the date/time of receipt of the message;
- **KSmsSize**, representing the message size (in bytes);
- **KSmsMode**, containing the encoding used to transmit the message;
- **KSmsBody**, which is the message itself.

The Asterisk dialplan processing can be used to store this message in a database, run any application, among others. However, the only action accepted by this channel is shutdown (Hangup), so this incoming call should be considered a special dialplan execution without audio streams or channel allocation.

## Channel "Khomp\_USSD"

This communication channel is used to receive USSD messages and create incoming links in Asterisk for each message received. This channel does not have any treatment or audio processing, and is called with one variable set:

- **KSmsBody**, which is the message itself.

The Asterisk dialplan processing can be used to store this message in a database, run any application, among others. However, the only action accepted by this channel is shutdown (Hangup), so this incoming call should be considered a special dialplan execution without audio streams or channel allocation.

## Channel "Khomp\_PR"

This communication channel is used to receive calls on devices passive recording (**KPR** and **KPR-300S**), creating incoming links on Asterisk for each incoming call. This channel allows only receiving audio captured the *linkby* not allowing both the transmission of audio signals as the control.

The Asterisk dialplan processing can be used to record data on this link in a database, perform some special application and/or some recording application (such as **Monitor**, **MixMonitor** or **KRecord**), among others. However, the only action accepted by this channel is shutdown (Hangup), so this should not be considered a common call.

## Management Interface (AMI)

The management interface AMI allows an external program to control the Asterisk and bonds managed by it. To facilitate this management of the channel Khomp offers the following facilities:

### List of commands

The AML commands allow the channel perform auxiliary functions and/or connections through an external program. The commands available are:

### KSendSMS

Send SMS messages through channels of the Khomp GSM devices. Required fields for this command are:

- **Device:** specifies the device to allocate (for more information, consult the documentation for item **action** of the application **KSendSMS**);
- **Destination:** sets the destination number to send the message;
- **Message:** Text (without quotes) to be sent to **destination**. If the **message** parameter is not provided, the SMS is sent without a message;

Optional fields:

- **Alert:** when set to true, sends SMS as alert message. Also called SMS Flash or SMS of class 0.
- **Linefeed:** When set to "true", it will replace the character "\n" (delivered as character '\ and character 'n' by AML) as linefeed. Available since version 4.3.2.
- **Confirmation:** when set to true, requests a confirmation message sent.
- **Encoding:** sets the encoding of the SMS, possible values are:
  - **7:** sends the message in GSM 7 bits encoding - default encoding
  - **binary:** sends the message in binary encoding
  - **16:** Sends the message in GSM 16 bits encoding. **Note: This encoding is only available for sending WAP type message;**
- **Port:** Sets the URL port number sent in WAP message. The default port is 2948;
- **Wap:** Requests to send a SMS message of the WAP PUSH type with URL<URL>. This message can be sent in 8 or 16 bit encoding. The types of WAP message are:
  - **Service Indication (SI):** Sends a link to the URL and a text in the message;
  - **Service Load (SL):** Sends only the link to the URL. The **message** field must be empty;
- **Async:** when set to true, sends the message asynchronously. (

This resource is available since version: **4.2.5**.

)

This command returns **Success**, message correctly sent by the modem, or **Error**, message is not sent and the reason detailed in **Message** field.

Examples:

```
action: KSendSMS
device: b0c0
destination: 88888888
message: Test

Response: Success
Message: Message sent

action: KSendSMS
device: b0
destination: 88888888
message: Test.

Response: Error
Message: No free channel found

action: KSendSMS
device: b0
destination: 88888888
message: Test
async: true

Response: Success
Message: SMS successfully queued

Event: KSendSMSResponse
```

```
Privilege: command,all
Response: Success
Message: Message sent
Message Reference: 9
Channel: B0C0
```

```
Action: KSendSMS
Device: b0c0
Destination: 88888888
Linefeed: true
Message: Test line1\nline2\nline3.
```

## KSendUSSD

Sends USSD messages through channels of the Khomp GSM devices. Required fields for this command are::

- **Device**: specifies the device to allocate (for more information, consult the documentation for the **resource** item of the **KSendUSSD** application);
- **Message**: contains the message to be sent.

This command returns **Success** or **Error**. In case of error, the reason why the message could not be sent is presented in the **Message** field.

Example:

```
Action: KSendUSSD
Device: b0c0
Message: *123#
```

```
Response: Success
Message: Message sent
```

## KHangup

Allows end calls forcing a send command directly to the device, using indices Send SMS messages through channels of the device of KGSM Khomp. Required fields for this command:

- **Device**: specifies the device to disconnect (for channel 0 of device 0, for example, would use up**B0C0**);
- **Index**: the call rate within the channel, used in**KGSM** channels.

## KDialOffHook

Dial over a Khomp channel while offhook. Required fields for this command:

- **Channel**: specifies the channel to be used (for channel 0 of device 0, for example, would use **B0C0**);
- **Number**: digits to dial.

## Event List

The events of AMI communicate channel events to allow connections to a program/application that performs the external management and/or control links in the system. The events offered are:

### Alarm

Reports alarm notification on the channel. Contains the following information:

- **Channel**: channel name (in format 'Khomp/BxCy') where the alarm occurred;
- **Alarm**: name of the alarm occurred in the link (see Chapter "Codes and meanings").

### AlarmClear

Notification relates to alarm channel. Contains the following information:

- **Channel**: channel name (in format 'Khomp/BxCy') where the alarm ceased.

## AnswerInfo

Reports detection service in the channel. Contains the following information:

- **Channel:** channel name (in format 'Khomp/BxCy') where the alarm occurred;
- **Info:** type name service detected (for details, see the detailed description of the variable KCallAnswerInfo).

## AntennaLevel

Reports changes in signal level of the GSM antenna, available on KGSM devices. Contains the following information:

- **Channel:** channel name (in format 'Khomp/BxCy') representing the modem;
- **Signal:** percentage of the signal, 0-99 (inclusive).

## OperatorRegistry

Reports the successful registration of the modem in a GSM operator. Available only on devices KGSM. Contains the following information:

- **Channel:** channel name (in format 'Khomp/BxCy') representing the modem;
- **Operator:** name of the operator.

## BranchOffHook

Reports that extension is off hook, available on devices KFXS. Contains the following information:

- **Channel:** channel name (in format 'Khomp/BxCy') which is off the hook.

## BranchOnHook

Reports that extension is on the hook, available on KFXS devices. Contains the following information:

- **Channel:** channel name (in format 'Khomp/BxCy') which is on the hook.

## CollectCall

Reports detection call collect. Contains the following information:

- **Channel:** channel name (in format 'Khomp/BxCy') where the alarm occurred.

## NewSMS

Reports that there is a new SMS message on the device, available on KGSM devices. Contains the following information:

- **Channel:** channel name (in format 'Khomp/BxCy') where the message was received;
- **From:** telephone number where the message was sent (it is provided by the carrier, and may also contain textual information);
- **Date:** Date and time of sending the SMS message by the source;
- **Size:** message size (in bytes);
- **Mode:** encoding used in post;
- **Message:** the body of the message sent.

NOTE: This event is only sent if there is a context for incoming calls SMS dialplan set in, otherwise the SMS message is kept in the SIM card, thus avoiding the loss of these messages.

## NewUSSD

Reports that there is a new USSD message on the device, available on KGSM devices. Contains the following information:

- **Channel:** channel name (in format 'Khomp/BxCy') where the message was received;
- **Message:** the body of the message received.

## KDisconnectionCause

Reports the disconnection cause of the call on a Khomp channel.

- **Channel:** channel name (in 'Khomp/BxCy' format);
- **OrigAddr:** originator number;
- **DestAddr:** destination number;
- **Cause:** number code of the disconnection cause;
- **Cause-txt:** text code of the disconnection cause;

## Gateway Interface (AGI)

To facilitate the use of the channel Khomp, AGI commands are provided for certain functions, listed below:

### KSENDSMS

An AGI command to send SMS messages through the channel, using the ISO-8859-1. The command has the following syntax:

```
kendsms <resource> <destination> <message>
```

Where the fields have the following meanings.

- **<resource>** Defines the channel to be allocated (for details, see item "feature" of the application documentation "KSendSMS");
- **<destination>** Is the target phone, where the message is sent;
- **<message>** Contains the body of the message being sent, without quotation marks;

The return code from this command, if successful, is:

```
200 result = 1
```

In case of failure, the return is:

```
200 result = 0 (<error-code>)
```

Where **<error-code>** is a code defined by the API K3L, for example:

```
200 result = 0 (kgccUnallocatedNumber)
```

If the destination number is nonexistent. For details, please consult the following chapter, "Codes and meanings."

### KSENDUSSD

An AGI command to send USSD messages through the channel. The command has the following syntax:

```
ksendussd <resource> <destination> <message>
```

Where the fields have the following meanings.

- **<resource>** Defines the channel to be allocated (for more information, consult the documentation for the "resource" item of the "KSendUSSD" application);
- **<message>** Contains the body of the message being sent, without quotation marks;

The return code from this command, if successful, is:

```
200 result = 1
```

In case of failure, the return is:

```
200 result = 0 (<error-code>)
```

Where **<error-code>** is a code defined by the API K3L, for example:

```
200 result= 0 (30,No network service)
```

If the service is nonexistent. For details, please consult the following chapter, "Codes and meanings."

## Asterisk Channel Tech API

To facilitate the sms sending through an external interface, khomp has implemented send\_text function (part of the struct ast\_channel\_tech khomp\_sms\_tech)(

This resource is available since version: **4.3.1**.

)

The use is made as follows:

```
chan_tech = ast_get_channel_tech("Khomp_SMS");
```

```
chan_tech->send_text(NULL, texttosend)
```

where:

**texttosend**: Must comply with the following format: (device~number~message~options), where '~' was defined as default separator.

- **options**: 'a', to enable alert;
- **options**: 'c', to enable send confirmation;
- **options**: 'e16', to enable 16 bits encoding message;

Ex:

```
chan_tech->send_text (NULL, "b00c00~99887766~test~ace16")
```

Ex2:

```
chan_tech->send_text (NULL, "b00~99887766~teste");
```

## Multiparty on KGSM boards

(This feature is **deprecated** since version **4.0**.);

This feature is deprecated since version **4.0**

Starting from version **3.0**, the Khomp channel driver - combined with the **KGSM** boards - supports the use of **Multiparty** features, allowing up to five conference between remote participants (operator dependent), plus the audio channel of the board (local participant) - as well as various other applications that use the call waiting, call hold, or conference features.

## Prerequisites

To use the **Multiparty** feature in **KGSM** boards and devices, it is necessary to:

- Enable **Enable Call Waiting** configuration in **k3lconfig**, in the GSM resources section;
- Use a **SIM** card whose conference feature is active at the operator;



- Have a good level of signal.

## Basic concepts

The operation mode of calls when Multiparty happens from the following rules:

1. All new incoming call should be (one of the following):
  - Answered as an active call, so that the audio can be handled by an **IVR** or by a conference;
  - Discarded without being answered, to keep the current calls in the current state;
2. There cannot be more than one active or held call without these calls being in a conference;
3. There can be only one conference during the course of the calls.

That is, the following settings are possible:

Number of calls		1	2	3	4	5	6
Active		0	1	0	1	N	1
On hold		0	0	1	1	1	N

Being **N** a number larger than "1" (one) and less than or equal to "5" (five). Starting from the second incoming call, the answered calls already can be placed on conference.

Example:

- Considering this scenario, on the situation of answering a third call, already having a call on hold and another active: one should put all active calls on hold (joining them on conference and then placing the conference on hold), or disconnect the active call. After answering the third call, the call can be all joined in the conference together.

## Incoming calls

### Contexts for additional calls

Upon receiving a new call in a KGSM the channel with **Multiparty** feature, the same settings (specified in the configuration file) and same search criteria are used.

However, when an active call or held call already exists on a particular channel and a new call comes in, another kind of search is made first, appending suffix "-waiting" to the configured contexts for the KGSM channel.

For example, consider the following excerpt from a configuration file:

```
context-gsm-gsm = khomp-DD
context-gsm-alt = khomp-gsm
```

An incoming call on the board 0 will trigger a search for contexts in the following order:

```
khomp-gsm-00
khomp-gsm
```

In the case of a second incoming call on this same board, with the same configuration, on a channel that already has one call, the search will be performed in the following order:

```
khomp-gsm-00-waiting
khomp-gsm-waiting
khomp-gsm-00
khomp-gsm
```

This allows the user to create different contexts for situations where a conference will be initiated and where it will be joined.

## The *waiting* extension

When a new call is detected by the channel, and there is a call / conference being addressed by the *activedialplan*, you can not answer this new call without putting the call on hold first.

To keep this scheme under the control of the programmer dialplan **when a new call is detected by the channel in this situation, there will be a search in the current context of execution of dialplan by the extension waiting, making a Goto for this extension if it exists.**

This allows it to be scheduled a specific action (put the conference on hold, for example) and treats a new call in a controlled manner.

For details, see the example dialplan at the end of this document.

## New channels

### Khomp\_MPTY

- Usage: logically represents a multiparty conference, running in a specific context.

Such a channel is created automatically when any of the calls run the application **KGsmMultiparty**, and the current call on hold and redirected to the audio channel that represents the conference.

### Khomp\_Wait

- Usage: logically represents an incoming call that is on standby.

Such a channel is created automatically when there is a context defined by the input option **context-gsm-wait** in the configuration file, and a new incoming call arrives on a system where there is already an active call in progress.

This channel does not have any treatment or audio alerts, running just to allow execution of external commands or the shutdown of the call.

You can force a shutdown of the current call running the application **Hangup** with the parameter **21** (call reject), as follows:

```
[Khomp-waiting-00-01]
exten => s,1,Hangup (21)
```

The origination of the call can be verified using the variable **\$(CALLERID(num))**.

## New applications

### KGsmMultipartyStart

- Usage: invoked from a call you want to start a conference, or from a channel conference already created, to encompass all the current channels in the same conference.

Receives as a parameter:

1. A context (optional)
2. An extension (optional)
3. Priority (optional).

The pair context/extension defines where to start executing the dialplan for the conference. If not specified, the channel **Khomp\_MPTY** is not created, and it will not be possible to perform any flow control for the conference. The extension default, if omitted, is **"s"**. If the channel **Khomp\_MPTY** has already been initialized, its execution flow will be diverted to the triple "context/extension/priority". If the context is omitted, execution will resume from where it was interrupted.

Examples:

```
; Take a call and creates a channel for the conference.
[empty]
exten => s, 1, WaitForSilence (99999)
```

```
[Khomp-gsm]
exten => s, 1, Answer ()
exten => s, n, KGsmMultipartyStart (mpty)
```

## KGsmMultipartyStart2

- Usage: works the same way as the application **KGsmMultipartyStart**, but allows specification of additional implementation options.

Receives as a parameter:

1. Runtime options (optional)
2. A context (optional)
3. An extension (optional)
4. Priority (optional).

The following implementation options can be used:

- **f**: Specifies that the first channel to enter the **Multiparty** call should be the owner of the conference - when this channel is disconnected, all other channels are also disconnected.  
**IMPORTANT**: All calls that are subjected to this ownership should invoke **KGsmMultipartyStart** with the parameter **f** set. Invocations which do not use this parameter will not be disassociated from the conference when the owner hangs up.
- **G**: If the channel conference has already been created, ignores the parameters context/extension/ priority, resuming the execution for the channel conference **Khomp\_MPTY** to the point of where it was interrupted (if it was already created);
- **X(n)**: Followed by the numeric parameter **N**, excludes the participant **N** from the conference, keeping it on hold.

The processing of other parameters follow the same logic of the application **KGsmMultipartyStart**.

Examples:

```
; Take a call and creates a channel for the conference.
[Mpty]
exten => s,1,WaitForSilence (99999)

[Khomp-gsm]
exten => s,1,Answer ()
exten => s,n,KGsmMultipartyStart2 (f,mpty)
```

## KGsmMultiparty

- Usage: invoked from a call you want to start a conference, or from a channel conference already created, to encompass all the current channels in the same conference.

Receives as a parameter:

1. A context (optional)
2. An extension (optional)
3. Runtime options (optional).

This application is deprecated, please use application **KGsmMultipartyStart** instead.

## KGsmMultipartyBreak

- Usage: invoked from a conference, breaking the conference by placing the current owner of this running again, and the other participants on hold - interrupting the execution of the conference channel **Khomp\_MPTY**, if one exists.  
The implementation of this channel when the owner returns the conference to run the application **KGsmMultiparty**.

By default, the *application* means that the channel on the owner of the conference has resumed its execution - that is, the current run of application **KGsmMultiparty** by the developer conference will return.

However, you can specify a new location dialplan where such implementation must continue with the following parameters:

1. A context (optional)
2. An extension (optional)
3. Priority (optional).

Examples:

```
; break the execution of the conference if the user press "#".
[empty]
exten => s,1,WaitExten (999)
exten => s,n,Goto (1)

exten => #,1,KGsmMultipartyBreak ()
exten => #,n,Goto (s,1)

; break the execution of the conference if the user press "#"
; redirecting the owner to the context "break", exten 's',
; priority "1" (same syntax as Goto).
[empty]
exten => s, 1, WaitExten (999)
exten => s, n, Goto (1)

exten => #, 1, KGsmMultipartyBreak (break, s, 1)
exten => #, n, Goto (s, 1)
```

## KGsmMultipartyOwner

Invoked from a call or conference, the conference sets the owner to a new value, or disable the feature completely.

Receives as a parameter:

1. The number of the new owner.

As the number of new owner, may be used reserved words "none" for any owner, or "detach" to disable the owner of the conference.

Examples:

```
; Adjusting the owner of the conference for the second participant (1)
exten => s, 1, KGsmMultipartyOwner (1)

; Temporarily disabling the owner of this conference call.
exten => s, 1, KGsmMultipartyOwner (none)
```

## KGsmDial

- Usage: invoked from an active conference call or in order to make a call to an external number using the same *chipfor* the current call.

Receives, as parameters:

1. The number to dial;
2. Dialing options (optional)
3. A context (optional)
4. An extension (optional).

The call / conference is now on hold until the number is disconnected, or to be served and placed on hold (to invoke the *application* **KGsmHold**), or to be serviced and placed in conference (invoke application **KGsmMultiparty**).

How dialing options, the following modifiers are accepted:

- **f**: When the context and parameters extension is not provided, whenever a new call comes into the conference to be satisfied (through the application **KGsmMultiparty**). If the parameter **f** is passed to the application **KGsmDial**, it will be passed on to the application **KGsmMultiparty**, making the new as the owner has called the first conference

participant;

- **M**: Resume dialplan execution when the original channel that invoked the application become the active channel again. If this option is not specified, the application also resumes dialplan execution if the conference channel becomes the active channel;
- **o**: If the connection is met with success, this will be the owner of the conference;
- **T**(timeout): Sets timeout`timeout` as maximum time to connect the call.

In return the call, two variables are set:

- **KGsmDialStatus**  
has values "**SUCCESS**", "**FAILURE**", "**ABORTED**" and "**TIMEOUT**", indicating - respectively - the call is successfully completed, if a fault was detected by the operator or by the modem, if the call was aborted, or if there was a timeout.
- **KGsmDialCode**  
has the specific error code of the call, and is set when the result of the variable **KGsmDialStatus** equals to "**FAILURE**". The possible values can be found in this document, in section Call codes.

Examples:

```
, Calling number 99887766 and placing it in conference
, immediately after the call.
[empty]
exten => s,1,KGsmDial(99887766)

, Calling number 99887766 and placing it in conference
, immediately after the call, using the "owner" feature.
[empty]
exten => s,1,KGsmDial(99887766,f)
```

## KGsmHold

- Usage: invoked from a call or conference, puts the active conference call or on hold, given incoming call is not met (if any) or re-starting the implementation of an earlier conference call or put on hold. If there is no incoming call waiting to be served, or any other call/conference call on hold, the application does not perform any action.

This application does not receive parameters.

## New commands

### khomp calls show

- Usage: Shows the GSM channels and their associated calls.

Receives as a parameter, a device number that is used to limit the list of channels belonging to this device.

## New in AMI

### New commands

#### KHangup

Hang up a call, sending a command directly to the board.

- Fields:
  - Channel: Channel name (Khomp / BnCy), and **x** the card, **y** the channel.
  - Index: Index of call.

### New events

#### HoldStart

Indicates that a particular call is on hold.

- Fields:
  - Channel: Channel name (Khomp / BnCy-z) where the event occurred, and **x** the card, **y** the channel, and **z** the index call.

**HoldStop**

Indicates that a particular call is no longer on hold.

- Fields:
  - Channel: Channel name (Khomp / BnCy-z) where the event occurred, and **x** the card, **y** channel and **z** the index call.

**MptyStart**

Indicates that a particular call is in conference.

- Fields:
  - Channel: Channel name (Khomp / BnCy-z) where the event occurred, and **x** the card, **y** the channel, and **z** the index call.

**MptyStop**

Indicates that a particular call is no longer in the conference.

- Fields:
  - Channel: Channel name (Khomp / BnCy-z) where the event occurred, and **x** board, **y** the channel, and **z** the index call.

**Some dialplan examples**

Below are some examples of dialplan conference using Multiparty feature of the channel.

**Simple Conference**

This example waits for incoming calls, and puts them in the conference, playing warning messages to users as new connections are received.

- khomp.conf

```
context-gsm-call = khomp-gsm
```

- extensions.conf

```
[default]
exten => s,1,Playback(fpm-calm-river)
exten => s,n,Goto(1)

exten => waiting,1,NoOp(waiting call)
exten => waiting,n,Playback(conf-lockednow)
exten => waiting,n,KGsmHold()
exten => waiting,n,Playback(conf-unlockednow)
exten => waiting,n,Goto(${WAITINGEXTEN}|1)

[khomp-gsm]
exten => s,1,NoOp(first call)
exten => s,n,Answer()
exten => s,n,Wait(2)
exten => s,n,Playback(hello-world)
exten => s,n,KGsmMultiparty(default)
exten => s,n,Playback(vm-goodbye)

[khomp-gsm-waiting]
exten => s,1,NoOp(other calls)
exten => s,n,Answer()
exten => s,n,Wait(2)
```

```

exten => s,n,Playback(conf-enteringno)
exten => s,n,Playback(digits/0)
exten => s,n,KGsmMultiparty()

```

## Advanced Conference

This example waits for incoming calls, and placing them in the conference, allowing the owner (the first party) to perform various actions - to dial a new participant and place it in a conference, talk privately with the same <sup>[1]</sup>, transfer ownership to another conference participant, or even excluded from the conference and allow it to continue until the last participant off.

1. ? If the restrictions of the GSM protocol are followed, as can be verified on the table of valid states listed earlier in this chapter.

```

;;[general]
;;autofallthrough=no
;;clearglobalvars=yes

;; Context for default calls (incoming).
[khomp-gsm]
exten => s,1,Set (GLOBAL (PHONE_${CHANNEL:6:4})=00000000)

;; WARNING: If using Local, always leave a playback here (even if just lms of silence).
;; WARNING: This is *REQUIRED* to Local optimize itself out of the way and allow KGsm*
;; WARNING: applications to access the underlying Khomp channel.
exten => s,n,Playback(hello-world)

exten => s,n(mpty),KGsmMultiparty(khomp-mpty|s|f)
exten => s,n(disas),DISA(no-password|disa-mpty)
exten => s,n,Goto(disas)
exten => s,n,Hangup()

;; Context for dialed calls using KGsmDial, in private mode.
[khomp-gsm-private]
exten => s,1,KGsmMultiparty(||fX(1))
exten => s,n,Goto(khomp-mpty,s,wait)

;; Context for reading digits.
[disa-mpty]
exten => _1X.,1,Set (GLOBAL (PHONE_${CHANNEL:6:4})=${EXTEN})
exten => _1X.,n,Goto(khomp-gsm|s|mpty)

exten => _[23456789]XXXXXXX,1,Set (GLOBAL (PHONE_${CHANNEL:6:4})=${EXTEN})
exten => _[23456789]XXXXXXX,n,Goto(khomp-gsm|s|mpty)

exten => _0XX[23456789]XXXXXXX,1,Set (GLOBAL (PHONE_${CHANNEL:6:4})=${EXTEN})
exten => _0XX[23456789]XXXXXXX,n,Goto(khomp-gsm|s|mpty)

exten => *,1,Set (GLOBAL (PHONE_${CHANNEL:6:4})=${EXTEN})
exten => *,n,Goto(khomp-gsm|s|mpty)

exten => i,1,Playback(beeperr)
exten => i,n,Goto(khomp-gsm|s|disa)

exten => t,1,Playback(beeperr)
exten => t,n,Goto(khomp-gsm|s|disa)

exten => T,1,Playback(beeperr)
exten => T,n,Goto(khomp-gsm|s|disa)

;; Context where Khomp_MPTY channel runs.
[khomp-mpty]
exten => s,1,NoOp()
exten => s,n(wait),WaitExten()
exten => s,n,Goto(wait)
exten => s,n,Hangup()

exten => i,1,Playback(beeperr)
exten => i,n,Goto(s|wait)

exten => t,1,Playback(beeperr)
exten => t,n,Goto(s|wait)

exten => T,1,Playback(beeperr)
exten => T,n,Goto(s|wait)

exten => *1,1,KGsmMultipartyBreak()
exten => *1,n,Set (PHONE=${GLOBAL (PHONE_${CHANNEL:11:4})})
exten => *1,n,Gosub(mpty-action,s,1)
exten => *1,n,Goto(s|wait)

exten => *2,1,KGsmMultipartyBreak()
exten => *2,n,Set (PHONE=${GLOBAL (PHONE_${CHANNEL:11:4})})
exten => *2,n,Gosub(mpty-private,s,1)
exten => *2,n,Goto(s|wait)

;; Multiparty context for private dialed calls.
[mpty-private]
exten => s,1,NoOp(GOT PHONE ${PHONE})
exten => s,n,KGsmDial(${PHONE}|of|khomp-gsm-private)
exten => s,n,WaitExten(9999)

```

```

exten => #1,1,KGsmMultiparty(||f)
exten => #1,n,Return()

exten => #2,1,KGsmMultipartyOwner(2)
exten => #2,n,KGsmMultiparty(||f)
exten => #2,n,Return()

;; Multiparty context for ordinary dialed calls.
[empty-action]
exten => s,1,NoOp(GOT PHONE ${PHONE})
exten => s,n,Goto(${PHONE},1)

exten => _XX.,1,KGsmDial(${PHONE}|of|khomp-gsm)
exten => _XX.,n,Return

exten => *,1,KGsmMultipartyOwner(none)
exten => *,n(play),Playback(vm-goodbye)
exten => *,n,Wait(15)
exten => *,n,Goto(play)

```

## Compatibility with DAHDI in Dial String

This resource is available since version: **3.1**.

The DAHDI configuration is grouped by *Spans* that can be devices/links. *Spans* are used as logical groupings of channels and can be configured in groups manually or automatically. The channels of all devices are set/displayed in a single sequence starting with 1 and without the device distinction.

## Behavior compatible with DAHDI

Compatibility with the DAHDI is performed as follows:

- Dial Application:
  - Dial(Khomp/<channel#>[c|r<cadance#>|d]/<extension>)
  - Dial(Khomp/(g|G|r|R)<group#(0-63)>[c|r<cadance#>|d]/<extension>)
  - Dial(Khomp/i<span#>/<extension>)
- Groups options:
  - g - allocation of channels groups with ascending search
  - G - allocation of channels groups with descending search
  - r - allocation of channels groups with ascending *round robin* search
  - R - allocation of channels groups with descending *round robin* search
- General options:
  - i - ISDN channel restriction by *Span*. **(Accepted but not used)**
  - c - waits for DTMF digit to confirm answer. **(Accepted but not used)**
  - r<cadance#> - sets call cadence
  - d - forces support capacity on ISDN/SS7 call. **(Accepted but not used)**
- To enable compatibility with DAHDI is necessary to set the **dial-string-like-dahdi** option in

**khomp.conf** configuration file to **yes**. In addition, this setting changes the display of some of the Khomp channel commands.

- The groups serve as the DAHDI logical groupings, and the DAHDI defaults groups are automatically configured.
- The cadences on DAHDI style are indicated by numbers in the **khomp.conf**.

## Dialstrings examples

```

Dial(Khomp/1/2927)
Dial(Khomp/2r1/2927)
Dial(Khomp/G1/32625644)
Dial(Khomp/R1r2/99991234)

```



# Use of additional patches

For some resources provided by the channel - as in the detection and reporting of attendance by the mailbox and answering machine - you may need the use of patches for Asterisk®, designed to extend the capabilities of this software to better meet the needs of the Khomp channel.

These patches are in the distribution of the source code of the channel Khomp in a directory called **patches**, where each directory contains patches to one or more versions of Asterisk®, and documentation files explaining its functionality.

To apply these patches, you should follow these basic procedures:

- Unpack the source code of Asterisk:

```
$ tar -zxvf asterisk-1.x.y.tar.gz
```

- Switch the current directory to the directory of the source code of Asterisk:

```
$ cd asterisk-1.x.y
```

- Run the command **patch** as described below:

```
$ patch -p0 < patch-xyzw.diff
```

- Recompile Asterisk and, if indicated in the documentation, the Khomp channel too.

# Codes and meanings

This chapter presents the codes present in the channel Khomp and their meanings, used in both events as in the AMI console commands:

## Channel state

Reflect the state of the channel on the device. In the case of E1 links, the state may have one or more of the following:

- **Free**: the channel is free;
- **Busy**: the channel is not free (or occupied, or failure);
- **Outgoing**: the channel has an output connection;
- **Incoming**: the channel has an input connection;
- **Locked**: the channel is blocked;
- **Outgoing Lock**: The channel is blocked for outgoing calls;
- **Local Fail**: The channel has a fault (at this point);
- **Incoming Lock**: the channel is blocked for incoming calls;
- **Remote Lock**: there is a remote lock (at the other end) in this channel.

In the case of a FXS channel, the state is defined by one of these values:

- **On Hook**: the phone connected to this channel is on-hook or disconnected;
- **Off Hook**: the phone connected to this channel is off the hook;

- **Ringing:** the channel is being called;
- **Failure:** the channel is in failure due to communication problems between the central and the plate.

In the case of a GSM channel, the state is defined by one of the following values:

- **Idle:** the channel is free and available for calls;
- **Call In Progress:** the channel is busy on a call;
- **SMS In Progress:** the channel is busy sending / receiving SMS messages;
- **Modem Error:** an error occurred communicating with the modem channel;
- **SIM Card Error:** The SIM card is not present or is not inserted / detected correctly;
- **Network Error:** an error occurred while communicating with the network;
- **Not Ready:** The modem is initializing the channel.

And in the case of an FXO channel, the states are as follows:

- **Disabled:** the channel is disabled;
- **Enabled:** the channel is enabled.

## Call state

Defines the logical state for each channel, which can be:

- **Free:** the channel is free;
- **Incoming:** the channel is receiving a call;
- **Outgoing:** the channel is making a call;
- **Failure:** the channel is in fault.

## Asterisk call states

Directly reflects the call state controlled by Asterisk, which can be:

- **Down:** Channel disconnected;
- **Reserved:** Channel allocated for subsequent outgoing link;
- **Offhook:** Channel allocated for outbound connection;
- **Dialing:** Canal in the dialing process;
- **Ring:** Connecting incoming calling;
- **Ringing:** Connecting outbound calling;
- **Ongoing:** Connection;
- **Busy:** Connection to outgoing channel busy;
- **Offdial:** Not used by the channel Khomp (*offhookdialing*);
- **Mute:** Not used by the channel Khomp (*mute*).

## GSM Codes

The following numeric codes are reported:

### SMS codes (SMS causes)

```

1      Unassigned number
8      Operator determined barring
10     Call barred
21     SMS transfer rejected
27     Destination out of service
28     Unidentified subscriber
29     Facility rejected
30     Unknown subscriber
38     Network out of order
41     Temporary failure

```

```

42 Congestion
47 Resources unavailable
50 Facility not subscribed
69 Facility not implemented
81 Invalid SMS transfer reference value
95 Invalid message
96 Invalid mandatory information
97 Message type non existent
98 Message not compatible with SMS protection state
99 Information element non existent
111 Protocol error
127 Interworking
128 Telematic interworking not supported
129 SMS type zero not supported
130 Cannot replace SMS
143 Unspecified TPPIID error
144 Alphabet not supported
145 Message class not supported
159 Unspecified TPDCS error
160 Command cannot be actioned
161 Command unsupported
175 Unspecified TP command error
176 TPDU not supported
192 SC busy
193 No SC subscription
194 SC system failure
195 Invalid SME address
196 Destination SME barred
197 SM rejected duplicate SM
198 TPVPP not supported
199 TPVP not supported
208 SIM SMS storage full
209 No SMS storage capability in SIM
210 Error in SMS
211 Memory capacity exceeded
213 SIM data download error
255 Unspecified error
300 Phone failure
301 SMS service reserved
302 Operation not allowed
303 Operation not supported
304 Invalid PDU mode parameter
305 Invalid text mode parameter
310 SIM not inserted
311 SIM PIN necessary
312 Phone SIM PIN necessary
313 SIM failure
314 SIM busy
315 SIM wrong
320 Memory failure
321 Invalid memory index
322 Memory full
330 SMSC address unknown
331 No network service
332 Network timeout
500 Unknown error
512 Network busy
513 Invalid destination address
514 Invalid message body length
515 Phone is not in service
516 Invalid preferred memory storage
517 User terminated

```

## Call codes (call causes)

```

1 Unallocated number
3 No route to destination
6 Channel unacceptable
8 Operator determined barring
16 Normal call clear
17 User busy
18 No user responding
19 No answer from user
21 Call rejected
22 Number changed
26 Non Selected user clear
27 Destination out of order
28 Invalid number format
29 Facility rejected
30 Response status enquiry
31 Normal, unspecified
34 No circuit channel available
38 Network out of order
41 Temporary failure
42 Switch congestion
43 Access information discarded
44 Requested channel unavailable
47 Resource unavailable
49 QoS unavailable
50 Request facility not subscribed
55 Call barred with UG
57 Bearer capability not authorized

```

```
58 Bearer capability not available
63 Service not available
65 Bearer capability not implemented
69 Request facility not implemented
70 Only restricted bearer capability available
79 Service not implemented
81 Invalid call reference value
82 User not member of UG
88 Incompatible destination
91 Invalid transit network selected
95 Invalid message
96 Missing mandatory information element
97 Message type not implemented
98 Message incompatible with state
99 Information element not implemented
100 Invalid information element
101 Message incompatible with state (2)
102 Recovery on timer expiry
111 Protocol error
127 Interworking
```

## General codes (mobile causes)

```
0 Phone failure
1 No connection to phone
2 Phone adaptor link reserved
3 Operation not allowed
4 Operation not supported
5 Phone SIM PIN required
6 Phone FSIM PIN required
7 Phone FSIM PUK required
10 SIM not inserted
11 SIM PIN required
12 SIM PUK required
13 SIM failure
14 SIM busy
15 SIM wrong
16 Incorrect password
17 SIM PIN2 required
18 SIM PUK2 required
20 Memory full
21 Invalid index
22 Not found
23 Memory failure
24 Text string too long
25 Invalid character in text string
26 Dial string too long
27 Invalid character in dial string
30 No network service
31 Network timeout
32 Network not allowed
33 Command aborted
34 Number parameter instead of text parameter
35 Text parameter instead of number parameter
36 Numeric parameter out of bounds
37 Text string too short
40 Network PIN required
41 Network PUK required
42 Network subset PIN required
43 Network subset PUK required
44 Network service provider PIN required
45 Network service provider PUK required
46 Corporate PIN required
47 Corporate PUK required
60 SIM Service option not supported
100 Unknown
103 Illegal MS #3
106 Illegal MS #6
107 GPRS service not allowed #7
111 PLMN not allowed #11
112 Location area not allowed #12
113 Roaming not allowed #13
132 Service option not supported #32
133 Registration service option not subscribed #33
134 Service option temporary out of order #34
147 Long context activation
148 Unspecified GPRS error
149 PDP authentication failure
150 Invalid mobile class
151 GPRS disconnection TMR active
256 Too many active calls
257 Call rejected
258 Unanswered call pending
259 Unknown calling error
260 No phone number recognized
261 Call state not idle
262 Call in progress
263 Dial state error
264 Unlock code required
265 Network busy
266 Invalid phone number
267 Number entry already started
```

```
268 Canceled by user
269 Number entry could not be started
280 Data lost
281 Invalid message body length
282 Inactive socket
283 Socket already open
```

# Troubleshooting

In this section, errors and their most common solutions are presented.

## Error during installation of kernel module

During installation of the *channel* of Khomp may occur the following messages:

```
K3L: WARNING: Unable to find the module for [...]
```

or

```
install: ***** THE KERNEL MODULE HAS NOT BEEN INSTALLED: *****
install:
install: ** Please, untar the file kpdriver*.tar.gz located in: **
install: **           '/usr/src/khomp/'           **
install: **           then check the README.txt           **
install: ** for knowing how to proceed with the installation. **
```

In this case, you must compile the drivers manually to your system. Proceed to the item below for more information.

## Compiling the drivers and starting the services

Just follow to the directory **/usr/src/khomp**, unpack the file **""kpdriver\_AAAAMMDD\_XXXX.tar.gz""**, and follow procedures described in the file **README\_en.txt**.

After performing compilation and installation of the module, just load it into the system, configuring the devices and start the server processes Khomp.

To load the kernel driver, you must run the following command:

```
# /etc/init.d/khompdrv start
```

Use **KWebPortal** to configure the devices in Channel 4.0, and in previous versions you need to run the command:

```
# khompwizard
```

This will run a setup wizard that will ask the signalling used in the system, as well as other parameters of use of the boards.

If necessary configure other additional parameters in versions prior to 4.0 you can use the following command:

```
# k3lconfig
```

This configurator, in turn, shows all possible options for card configuration. The parameters that are not configured automatically assume the default values, and are compatible with most systems. More details about this program can be obtained from the section number '2'.

- **ATTENTION': To start Asterisk®, it is necessary that the Khomp device is configured and all modules are running (as shown above). If the device is not configured, Asterisk will not start.**

**If you want to run the system without the Khomp device, you need to configure Asterisk for it does not load the module Khomp. To do this, open the "/etc/asterik/modules.conf and add the line:**

```
noload => chan_khomp.so
```

Khomp Ind. e Com. Ltda. 1996-2015 :: Rua Joe Collaco, 253 :: Florianopolis/SC :: Brasil :: +55 48 3722-2900

When the

Khomp device is properly configured and modules khomp loaded (explained above), remember to remove this line from the file.

Finally, to load the server process in Channel 4.0, simply run the following command:

```
# k3lserver start
```

In versions prior to 4.0, simply run the following command:

```
# kserver start
```

After performing these procedures, the channel is already operational, and Asterisk can now be loaded.

## Setting up special parameters for audio or signaling

In Channel 4.0, the specific parameters of timing and/or signaling are configured in **KWebPortal** described in user's\_guide.

To set specific parameters of timing and/or signaling in versions prior to 4.0, you can use the program "k3lconfig": simply select the card you want, and options of the boards will be presented, divided into sections and subsections for easy access. It is not necessary to effect the configuration of all parameters: default values are assumed, if not configured. To adjust the signaling link, simply - after selecting the card - enter the "Options signaling, and then in" Signs of the line. " To choose a particular signaling, just use the direction keys (arrows) to select it, press 'space', and confirm the option by pressing 'Enter' on the "Confirm" button. Finally, to save the modified settings, just exit the program: it will display a window with options to save the changes made or not.

Please not that **the following options are required to be unchanged from the defaults**:

- Automatic echo cancellation;
- DTMF suppression;
- Automatic Gain Control (AGC).

These options are **controlled by channel** and should be **disabled** (the default configuration).

## Automatic load of services and kernel modules

If the loading of kernel module or Khomp services startup is not performed automatically at startup, you can perform this installation manually, creating links for the Khomp scripts in the system startup directory. In Channel 4.0, the scripts are: **/etc/init.d/khompdrv**, **/etc/init.d/klogserver**, **/etc/init.d/k3lserver**, **/etc/init.d/kqueryserver**, **/etc/init.d/kwebserver** and **/etc/init.d/kibs**. In versions prior to 4.0: **/etc/init.d/khompdrv** and **/etc/init.d/kserver**.

In the case of **Debian**-based distributions the script for loading kernel modules would be linked within the directory **/etc/rcS.d**, while the services loader would be linked within the directories **/etc/rc2.d**, **/etc/rc3.d**, **/etc/rc4.d**, **/etc/rc5.d** as follows:

In Channel 4.0:

```
# ln -s /etc/init.d/khompdrv /etc/rcS.d/S19khompdrv
# ln -s /etc/init.d/klogserver /etc/rc2.d/S13klogserver
# ln -s /etc/init.d/klogserver /etc/rc3.d/S13klogserver
# ln -s /etc/init.d/klogserver /etc/rc4.d/S13klogserver
# ln -s /etc/init.d/klogserver /etc/rc5.d/S13klogserver
# ln -s /etc/init.d/k3lserver /etc/rc2.d/S15k3lserver
# ln -s /etc/init.d/k3lserver /etc/rc3.d/S15k3lserver
# ln -s /etc/init.d/k3lserver /etc/rc4.d/S15k3lserver
# ln -s /etc/init.d/k3lserver /etc/rc5.d/S15k3lserver
# ln -s /etc/init.d/kwebserver /etc/rc2.d/S20kwebserver
# ln -s /etc/init.d/kwebserver /etc/rc3.d/S20kwebserver
# ln -s /etc/init.d/kwebserver /etc/rc4.d/S20kwebserver
# ln -s /etc/init.d/kwebserver /etc/rc5.d/S20kwebserver
# ln -s /etc/init.d/kibs /etc/rc2.d/S20kibs
# ln -s /etc/init.d/kibs /etc/rc3.d/S20kibs
# ln -s /etc/init.d/kibs /etc/rc4.d/S20kibs
# ln -s /etc/init.d/kibs /etc/rc5.d/S20kibs
# ln -s /etc/init.d/kqueryserver /etc/rc2.d/S20kqueryserver
# ln -s /etc/init.d/kqueryserver /etc/rc3.d/S20kqueryserver
# ln -s /etc/init.d/kqueryserver /etc/rc4.d/S20kqueryserver
```

```
# ln -s /etc/init.d/kqueryserver /etc/rc5.d/S20kqueryserver
```

In versions prior to 4.0:

```
# ln -s /etc/init.d/khompdrv /etc/rcS.d/S19khompdrv
# ln -s /etc/init.d/kserver /etc/rc2.d/S20kserver
# ln -s /etc/init.d/kserver /etc/rc3.d/S20kserver
# ln -s /etc/init.d/kserver /etc/rc4.d/S20kserver
# ln -s /etc/init.d/kserver /etc/rc5.d/S20kserver
```

Please check the rules of your distribution to initialize the services in accordance with what is expected by the start of it.

## Appendix

In this section, are useful information about the channel and related components.

### Arrangement of installed files

The directories created/modified in this facility are:

```
/etc/init.d/      - Startup scripts;

/etc/khomp/       - Firmware files and settings;

/etc/asterisk/    - Settings for Asterisk and Channel;

/usr/doc/khomp/   - Docs for the devices, Channel, and utilities;

/usr/sbin/        - Utilities and server processes;

/usr/lib          - Shared libraries;

/usr/lib/asterisk/modules/ - Channel module;

/var/log/khomp/   - Log directory for K3L and channel
                  (Channel 4.0);
/var/log/khomp2.2/ - Log directory for K3L and channel
                  (Channel 3.1);
/var/log/khomp2.1/ - Log directory for K3L and channel
                  (Channel 3.0);
/var/log/khomp2.0/ - Log directory for K3L and channel
                  (Channel 2.4);
```

The script **/etc/init.d/khompdrv** is responsible for loading modules **kpci9030.ko** and **kpex8311.ko** in the kernel, which should be carried out automatically at system startup. In case of problems, check the Troubleshooting section.

### Compatibility with Zaptel/DAHDI modules

In some cases, when the installation of Asterisk is made by some package manager, the Zaptel/DAHDI drivers can be loaded by default, even when not needed. If they are not used, it is suggested to be removed as a matter of compatibility - explained in more detailed below.

Here are the steps to check if the module is loaded and how to remove it:

- To find out if the modules are loaded:

```
# lsmod | grep zaptel
# lsmod | grep dahdi
```

- If any of the above command to return some module, meaning that at least one of the drivers is loaded. To remove it, run:

```
# rmmod zaptel
```

or

```
# rmmod dahdi
```

- You must certify that this module is not reloaded. On Debian this can be done by removing the package **zaptel**, or the package **dahdi**:

```
# apt-get remove zaptel
```

or

```
# apt-get remove dahdi
```

If necessary, you can use the Zaptel modules/DAHDI with Khomp devices, but this requires booting **first** modules of Khomp.

This is due to the device detection system implemented in the Zaptel/DAHDI drivers, where Khomp devices are incorrectly initialized by them. To prevent this incorrect detection, the Khomp drivers must be fully initialized by the time the Zaptel/DAHDI drivers are loaded.