Channel Khomp: User Guide

Sumário

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

These steps are described more fully below.

K3L API Configuration

This step is carried out in a semi-automated way using the **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, the program **k3lconfig** allows you to access all the available settings for each installed card. For more information about this program, check the **k3lconfig User's Guide**. For solving synchronization issues, check the Troubleshooting section for manual configuration of the boards.

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:

[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) (**Available since version: 3.0**);
- **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 boards (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 cards (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 cards (the default is empty - when set, follows the same substitution rules for **context-fxo**);
- **context-fxs**: Context for incoming connections on FXS cards (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 cards (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 boards (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);
- **context-pr**: Context for incoming connections on boards 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 an 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 an 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);
- **fxo-send-pre-audio**: When enabled ("yes") releases audio channel before the outgoing call connection boards KFXO (the default value is "yes");
- **fxo-callerid-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 board 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) (**Available since version: 3.0 (special package)**);

- **fxo-callerid-timeout**: Defines the maximum time for recognizing the originator number using the **CallerId** feature (default value is 2000 ms) (local option) (**Available since version: 3.0 (special package)**);
- **fxs-digit-timeout**: Defines the timeout, in seconds, between digits of a FXS board's extension (local option);
- **fxs-global-orig**: Start number for sequencial branch numbering in FXS cards that are not listed in the **[fxs-branches]** section (the numbering follows ascending order from board 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) **Available since version: 3.0 (special package)**);
- **has-ctbus**: Enable ("yes") or disables ("no") support for CT-bus on **CTI** boards (not available on **SPX** models) (local option);
- **ignore-letter-dtmfs**: Defines if the channel should ignore some uncommon DTMF digits detected by the board (A, B, C and D). However, if you need to pass those digits througth the board, 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 board), from -10 to +10 (local option);
- **kommuter-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" (**Available since version: 2.4.1**);
- **kommuter-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" (**Available since version: 2.4.1**);
- **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 happends 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);
- **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);
- **recording**: Sets whether automatic recording should occur (local option);
- **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 tranfer between Asterisk® and another PBX (via user signaling) (**Available since version: 3.0 (special package)**) (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) (**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) (**Available since version: 3.0 (special package)**) (local option);

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

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

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

For example, if two **KFXS-300 SPX** boards 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 board '*K0374* as number 200, the second as 201, and so one. The first branch from board 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 exemples.

[fxs-hotlines]

Sets hotlines for the KFXS based boards

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.

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*>]

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-b0l0+b1l0]
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 boards and devices Khomp, 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 boards, 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 board 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 card (three digits), and **SSSS**, which represents the serial board number (with four digits).

Examples for configuration entries:

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

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

; Sequential board 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 board 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 board 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 board **0** will have the following handling:
   - Calls to the extension 1234 for will be redirected to the second link on the first board (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 board **0** for the number **1111** will be redirected to the first link of the first board (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 cards

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

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

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

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

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

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

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 board.

- **NOTE**: In the **KGSM** board, 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**i** 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 channel Khomp and forwarded to Asterisk as a normal connection but no audio, which has some variables set with information received in the message - for more information on these variables, see the documentation of the variables of the channel. This context can also be modified in the same way as the above contexts.

The default value for this option follows:

```
context-gsm-sms-sms = khomp-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})
```

Contexts for Khomp_PR channels (KPR)

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

```
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 boards, 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 boards, links and channels, the following syntax is available (considering X, Y and Z as any numbers):

- b**X** -- search the channels on the board **X**, ascending order;
- b**X**L**Y** -- search channel on the link **Y** from **X** board, ascending order;
- b**X**c**Y** -- try to allocate channel **Y** from board **X**;
- b**X**c**Y**-**Z** -- search for channels starting from channel **Y** to channel **Z** (inclusive) of board **X**, ascending order;
- B**X**c**Y**-**Z** -- same as above, but descending order;
- s**X** -- search the channels on the board of serial number **X**, ascending order;
- s**X**L**Y** -- search channel on the link **Y** from board of serial number **X**, ascending order;
- s**X**c**Y** -- try to allocate channel **Y** from board of serial number **X**;
- s**X**c**Y**-**Z** -- search for channels starting from channel **Y** to channel **Z** (inclusive) from board of serial number **X**, ascending order;
- S**X**c**Y**-**Z**-- same as above, but descending order.

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

- r**X**- search branch numbered **X**;
- R**X**- equivalent to the above;

- r**X**-**Y**- search from branch **X** to **Y**, ascending order;
- R**X**-**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/*gOperadora1+gOperadora2/98891234)
```

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) boards**. 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.

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 boards 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);
- **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;
- **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). **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**);
- **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** boards, 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 boards, the only valid value for this option is **restricted**.

- **output_volume**: Sets the output volume of the link (ranges from -10 to +10);
- **usr_xfer**: Defines a group of DTMF digits to initiate a transfer between PBXes (using QSig or FXO FLASH, for instance);
- **uui**: 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:

- **KDropCollectCall**: 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) (**Available since version: 2.2**);
- **KR2SendCondition**: 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);
- **KR2GotCategory**: Adjusted by the channel when an incoming call is received, with the category number of the caller (only available on R2 signaling);
- **KR2GotCondition ***: 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 **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 signaling);
- **KISDNGotCause ***: 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 **before 1.8**, it **requires** a patch in Asterisk, or to adjust the "**parent**" Dial option accordingly on the outgoing call) (available only for ISDN signaling);
- **KCallAnswerInfo ***: 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 **before 1.8**, it **requires** a patch in Asterisk, or to adjust the "**parent**" Dial option accordingly on the outgoing call) (available only for digital signaling - E1 and GSM);
- **KSmsDelivered**: Adjusted by the channel when sending a SMS message with the application **KSendSMS**, saying whether the message was delivered successfully ("yes") or not ("no");
- **KSmsErrorCode**: Adjusted by the channel when sending a SMS message with the application **KSendSMS**, containing the error code that happened when sending the message.
  Until channel 2.4.1 this variable had a string describing the error name, since channel 3.0, it contains the error code number;
- **KSmsErrorName**: Adjusted by the channel when sending a SMS message with the application **KSendSMS**, contains the name of the error or "None" if there has been no error. (**Available since version: 3.0**);
- **KSmsType**: 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". (**Available since version: 3.0**);
- *"KSmsFrom'*: 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 ");
- **KSmsDate**: Adjusted for the input channel in the context of SMS messages, sets the date of sending the message (available on types "message" and "confirm");
- **KSmsSize**: Adjusted for the input channel in the context of SMS messages, contains the size (in bytes) of the received message (available on types "message" and "broadcast");
- **KSmsMode**: 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");
- **KSmsBody**: Adjusted for the input channel in the context of SMS messages, contains the text of the received message (available in types "message" and "broadcast");
- **KSmsDelivery**: 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") (**Available since version: 3.0**);
- **KSmsStatus**: Adjusted for the input channel in the context of SMS messages, contains the status of the message sent earlier (available on the type "confirm") (**Available since version: 3.0**);

- **KSmsSerial**: Adjusted for the input channel in the context of SMS messages, contains the serial number of the received message (available on the type "broadcast") (**Available since version: 3.0**);
- **KSmsPage**: Adjusted for the input channel in the context of SMS messages, contains the page number of the received message (available on the type broadcast") (**Available since version: 3.0**);
- **KSmsPages**: Adjusted for the input channel in the context of SMS messages, contains the total number of pages to be received (available in type "broadcast") (**Available since version: 3.0**);
- **KUserInfoDescriptor**: Sets/reports protocol descriptor of the User-to-User Information message (ISDN).
- **KUserInfoData**: Sets/reports data in the User-to-User Information message (ISDN).
- **KFaxSent**: Adjusted by the channel when sending FAX with **KSendFax** application, and determines whether the fax was successfully sent ("yes") or not ("No");
- **KFaxReceived**: Adjusted by channel when receiving FAX with **KReceiveFax** application, and determines whether the fax was successfully received ("yes") or not ("no");
- **KFaxResult**: Adjusted by the channel when sending or receiving FAX with the application **KSendFax** or **KReceiveFax** (respectively), and defines the result of execution.

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 **Ringing** 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 to**yes** 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
...
```

KR2SendCondition

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,KR2SendCondition (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. 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%.

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 (**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 board, 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 card in question, requesting a disconnection. Use with caution;
- **khomp calls show**: Shows states for calls, may also listing specific channels or boards (**Available since version: 3.0**);
- **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 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 (**Available since version: 3.0**);
- **khomp links errors**: Shows status of error counters for all channels, or for a board/link only;
- **Khomp reset links**: Sends a reset command for a specific **E1** of a particular *card*;
- **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 board);
        - **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 console log khomp**.
- **khomp log rotate**: Rotate log files from the channel;
- **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);
- **khomp log trace k3l**: Enables debugging low-level API K3L;
- **khomp log trace r2**: Enables debugging low-level signaling R2/MFC;
- **khomp record**: Enable audio recording through the Khomp channel;
- **khomp revision**: Shows version number and revision of the channel;
- **khomp select sim**: Select the SIM card, available on the boards KGSM (**Available since version: 2.4.1**);
- **Khomp send command**: Sends command API K3L directly to the board (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);
- **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**: Send an SMS message for a given number, using KGSM channels;
- **khomp summary**: Prints a summary of system boards and their features;
- **khomp kommuter**: Enables or disables kommuters connected via USB on this machine. Only accessible when the configuration "kommuter-activation" is set to "manual" (**Available since version: 2.4.1**);

- **khomp kommuter count**: Gets the amount of kommuters connected via USB on this machine (**Available since version: 2.4.1**).

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 signaling protocol QSig (Single Step Call Transfer) for boards configured with E1 ISDN signaling (ISDN), or use the FLASH command for FXO.***

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.
- **options**: Sets the transfer options to be used, which are:
    - **N**: Wait until the channel is disconnected.

"KRecord" application

Application to record audio through the boards Khomp. This application was created to use the recording feature in this Khomp boards, which can 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 (teste.wav)

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

"KStopRecord" application

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 boards of KGSM Khomp using modules and SIM cards in the board to do so. The syntax of the application is as follows:

```
KSendSMS (resource | destination | message)
```

Each field can be summarized in:

- **resource**: The following is a syntax identical to the allocation of channels Dial application, and defines what modem use;
- **destination**: Number where to send the message, may be preceded or succeeded by**!** to request a confirmation of transmission;
- **message**: Text (without quotes) to be sent to**destination**.

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." phone for "99887766" using the modem "1" (second modem) card "0":

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

- Sends "Test message." phone for "99887766" using the first free modem card "0", and checks the return shipment:

```
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}); only for channel 3.0
```

- Sends "Test message." phone for "99887766" using the first free modem card "0", or the first free channel board "1" (if no free channel at the first sign):

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

- Sends "Test message." phone for "99887766" using the first free modem card "0", requesting confirmation:

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

Application "KEchoCanceller"

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

Syntax updated application (**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 (**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 (**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 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 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 to select SIM card in khomp boards, and its syntax as follows:

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

Where the fields have the following meanings:

- **board**: Defines which card to send the command.
- **channel**: Defines which channel in the card to send the command.
- **sim_card**: The SIM card number to be selected.

Exemplos:

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

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**, that 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 boards passive recording (**KPR** and **KPR-300S**), creating incoming links on Asterisk for each incoming call. This channel allows only receiving audio captured the*link*by 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 commands allow the IFA channel perform auxiliary functions and/or connections through an external program. The commands available are:

KSendSMS

Send SMS messages through channels of the board of KGSM Khomp. Required fields for this command:

- '*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**: contains the message to be sent.
- **Confirmation**: When set to true, requests a confirmation message sent.

KHangup

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

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

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 boards KGSM. 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 boards 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 boards 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 boards KFXS. 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 card, available on boards KGSM. 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.

Gateway Interface (AGI)

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

KSENDSMS

AGI is a command for sending SMS messages through the channel, using the ISO-8859-1. The command has the following syntax:

```
ksendsms <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 (<código-de-erro>)
```

Where**<código-de-erro>** 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."

Multiparty on KGSM boards

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 then 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	Scenarios					
	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*active*dialplan, 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.
[mpty]
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 "#".
[mpty]
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).
[mpty]
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*chip*for 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.
[mpty]
exten => s,1,KGsmDial(99887766)

; Calling number 99887766 and placing it in conference
, immediately after the call, using the "owner" feature.
[mpty]
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 [1], transfer ownership to another conference participant, or even excluded from the conference and allow it to continue until the last participant off.

1. 2 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 1ms 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(disa),DISA(no-password|disa-mpty)
exten => s,n,Goto(disa)
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.
[mpty-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)
```

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 board. 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 (offhook*dialing*);
- **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 TPPID 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   TPVPF not supported
199   TPVP not supported
208   SIM SMS storage full
209   No SMS storage capability in SIM
210   Error in SMS
211   Memory capatity 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    Cancelled 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_2.0.0**XX**.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 boards and start the server processes Khomp.

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

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

To set up the boards, in turn, must 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 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 board is configured khomp and all modules are running (as shown above).*** *If the card is not configured, Asterisk will not start**.***

  ***If you want to run the system without the board Khomp, 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
  ```

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

Finally, to load the server process, 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

To set specific parameters of timing and/or signaling, 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** in 'k3lconfig' (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 a link for the scripts **/etc/init.d/khompdrv** and **/etc/init.d/kserver** in the system startup directory.

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:

```
# 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 boards, Channel, and utilities;

/usr/sbin/      - Utilities and server processes;

/usr/lib        - Shared libraries;

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

/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 Khomp with boards, but this requires booting **first** modules of Khomp.

This is due to the device detection system implemented in the Zaptel/DAHDI drivers, where Khomp boards 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.