

# Gaia Technical Report

Version 0.1

Javier Pérez Mayos  
TALP Research Center  
`javier.perez@tsc.upc.edu`

May 19, 2006

Copyright (c) 2005 Javier Pérez

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# Contents

<b>1</b>	<b>Gaia kernel</b>	<b>4</b>
1.1	Interfaces . . . . .	8
1.1.1	Common interface . . . . .	10
1.1.2	Speech recognizer interface . . . . .	10
1.1.3	Text translator interface . . . . .	10
1.1.4	Synthesizer interface . . . . .	10
1.1.5	User interface . . . . .	10
1.2	Events . . . . .	10
1.2.1	End of turn . . . . .	10
1.2.2	End of call . . . . .	11
1.2.3	Errors . . . . .	11
1.3	Event procedures . . . . .	12
1.3.1	EOC_TLK . . . . .	12
1.3.2	ULLST_ABORT . . . . .	13
1.3.3	UI_TLK_ABORT . . . . .	13
1.3.4	ASR_ABORT . . . . .	13
1.3.5	TTS_ABORT . . . . .	14
1.3.6	TTT_ABORT . . . . .	14
1.3.7	EOC_LST . . . . .	14
1.3.8	ASR_EOT and UI_EOT . . . . .	16
1.4	CommonInterface . . . . .	17
1.5	UserInterface . . . . .	18
1.5.1	UserInterface. Input thread . . . . .	18
1.5.2	UserInterface. Output thread . . . . .	18
1.6	AsrInterface . . . . .	20
1.6.1	AsrInterface. Input thread . . . . .	20
1.6.2	AsrInterface. Output thread . . . . .	20
1.7	TttInterface . . . . .	22
1.7.1	TttInterface. Input thread . . . . .	22
1.7.2	TttInterface. Output thread . . . . .	22
1.8	TtsInterface . . . . .	23
1.8.1	TtsInterface. Input thread . . . . .	23
1.8.2	TtsInterface. Output thread . . . . .	23

<b>2</b>	<b>Platform users</b>	<b>24</b>
2.1	Microphone . . . . .	24
2.1.1	Initial . . . . .	24
2.1.2	Normal . . . . .	25
2.2	DialogicUser . . . . .	25
2.2.1	Client mode . . . . .	26
2.2.2	Server mode . . . . .	26
2.2.3	Dialogic thread . . . . .	27
2.2.4	Input/ouput thread . . . . .	27
<b>3</b>	<b>GNU Free Documentation License</b>	<b>31</b>
1.	APPLICABILITY AND DEFINITIONS . . . . .	31
2.	VERBATIM COPYING . . . . .	33
3.	COPYING IN QUANTITY . . . . .	33
4.	MODIFICATIONS . . . . .	34
5.	COMBINING DOCUMENTS . . . . .	36
6.	COLLECTIONS OF DOCUMENTS . . . . .	36
7.	AGGREGATION WITH INDEPENDENT WORKS . . . . .	37
8.	TRANSLATION . . . . .	37
9.	TERMINATION . . . . .	37
10.	FUTURE REVISIONS OF THIS LICENSE . . . . .	37
	ADDENDUM: How to use this License for your documents . . . . .	38

# List of Figures

1.1	Translation platform . . . . .	5
1.2	Schematic diagram of the complete Gaia platform: text and audio clients, and ASR, SLT and TTS servers for the available languages. . . . .	6
1.3	Details of the Gaia kernel internals: client and server interfaces and core event handler. . . . .	7
1.4	Common interface . . . . .	9
2.1	Fields of the integer code . . . . .	25

# Chapter 1

## Gaia kernel

The translation platform is organized in an star way: all the different parts involved in the translation of speech from one language to another communicate with a central piece of software, the *core* of the platform.

As figure 1.1 shows, the core of the platform consists of several modules handling the communication with the different parts of the translation process (the servers performing the speech recognition, text translation, speech synthesis and user interfacing), and of a kernel module controlling everything.

From a *black box* perspective, each interface communicates through a socket with the corresponding server, sending commands to it and receiving the appropriate answer (if any) according to a pre-established protocol.

Internally, each interface reads commands from an input pipe, and sends its output (e.g. the recognized text given by the ASR) to another pipe, connected to the input of the next interface. In case of exceptional events or commands received, the interface can communicate directly with the kernel sending an *event* indicating the exceptional situation.

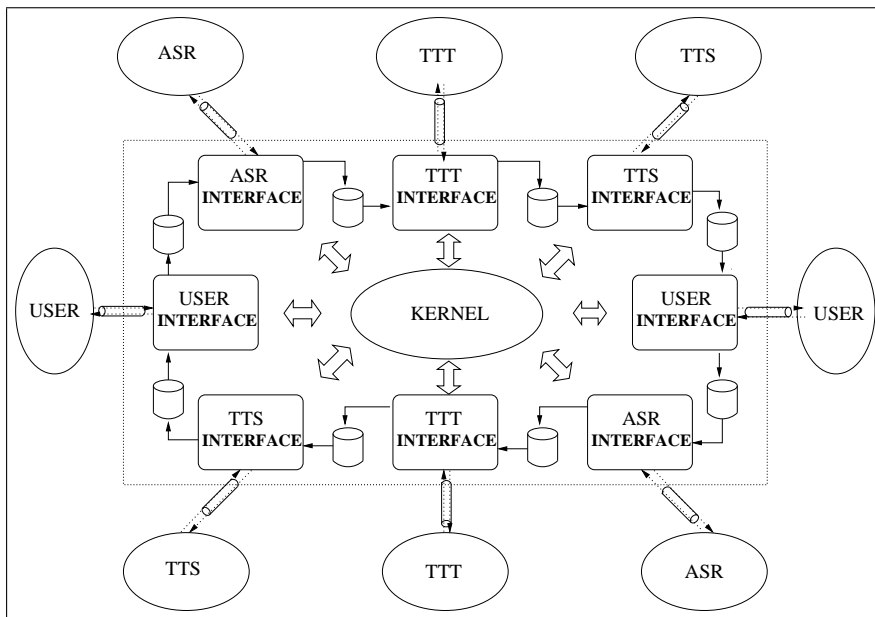


Figure 1.1: Translation platform

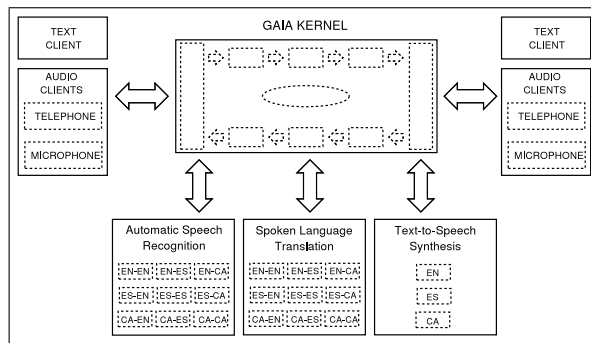


Figure 1.2: Schematic diagram of the complete Gaia platform: text and audio clients, and ASR, SLT and TTS servers for the available languages.

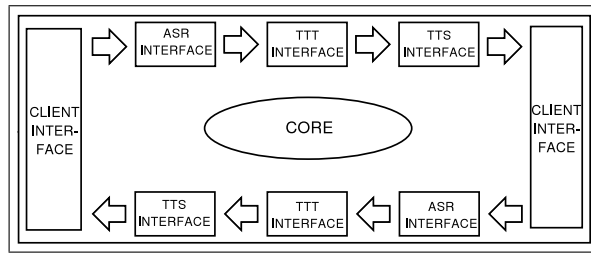


Figure 1.3: Details of the Gaia kernel internals: client and server interfaces and core event handler.

By *exceptional* events we mean a user hanging up the phone or marking the end of the turn by pressing a key, or an end of turn condition detected by the ASR server. As long as nothing of this nature happens, the normal operation of the platform is as follows:

1. The user talks
2. The *UserInterface* receives the audio data through the socket and writes it to its output pipe
3. The *AsrInterface* reads the audio data from its input pipe and sends it to the ASR server through the socket
4. When the ASR server sends the recognized text through the socket, the *AsrInterface* receives it and writes it into its output pipe
5. The *TttInterface* reads it and sends it to the TTT server through the socket
6. The TTT sends the translated text to the platform, and the *TttInterface* reads it and writes it into its output pipe
7. This text is read then by the *TtsInterface*, that sends it to the TTS server
8. The synthesized text is then received through the socket by the *TtsInterface*, and sent to the *UserInterface* using the output pipe
9. At last, the *UserInterface* reads the audio data from its input pipe and sends it to the user through the socket.

## 1.1 Interfaces

There are three different interfaces implemented so far, all of them sharing one common piece of code (*CommonInterface.cpp* and *CommonInterface.h*). Each interface consists of two threads, the input thread reading commands from the input FPIPE and sending commands through the socket to the server, and the output thread reading the answer from the socket, and sending the result to its output FPIPE. Figure 1.4 illustrates the interface operation.

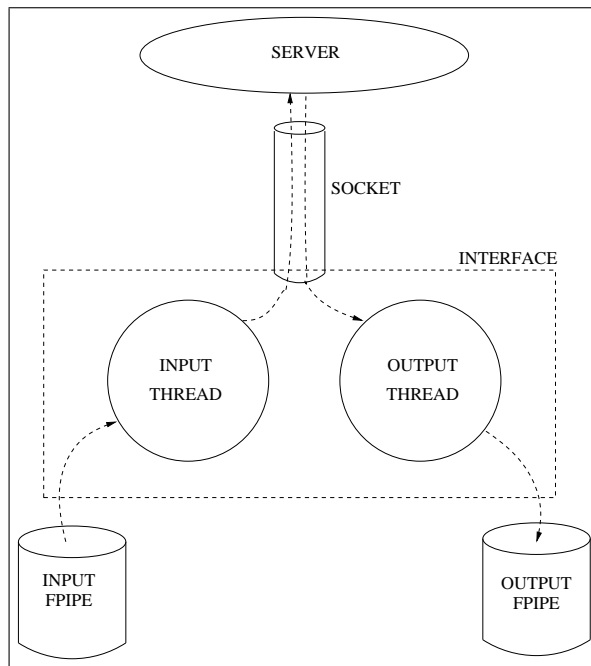


Figure 1.4: Common interface

### 1.1.1 Common interface

Files: *CommonInterface.h* and *CommonInterface.cpp*.

This is the code and functionality shared by all the different interfaces. It basically defines the stopping and aborting procedures, that are the same for all. Each particular interface has to define the start method, and define the way the different commands are dealt with.

The common interface also defines the state-machine and provides the functions to change and wait for states.

### 1.1.2 Speech recognizer interface

Files: *AsrInterface.h* defines the skeleton to be implemented by the actual interface. The file *iAsrInterface.cpp* provides an implementation.

### 1.1.3 Text translator interface

Files: *TttInterface.h* defines the skeleton to be implemented by the actual interface. The file *iTttInterface.cpp* provides an implementation.

### 1.1.4 Synthesizer interface

Files: *TtsInterface.h* defines the skeleton to be implemented by the actual interface. The file *iTtsInterface.cpp* provides an implementation.

### 1.1.5 User interface

Files: *UserInterface.h* defines the skeleton to be implemented by the actual interface. The file *iUserInterface.cpp* provides an implementation.

## 1.2 Events

### 1.2.1 End of turn

The end of turn can be handled by both the *UserInterface* and the *AsrInterface*. This way allows the users to finish their turns whenever they want (e.g. by pressing a key on the keypad) or the speech recognizer can detect the end of turn by using silence or maximum time talking measures.

If the *AsrInterface* receives an `{EOT_CMD, }` from the recognition server, it sends the command to the next interface through the output *FPIPE* and signals this exceptional condition to the kernel, by sending an `ASR_EOT` event (see 1.3.8 to understand how the kernel handles this event). In short, the kernel explicitly stops the talking *UserInterface*, and waits for all the others interfaces to stop (each interface stops when the `{EOT_CMD, c}` command is received).

When the `UserInterface` receives the `{EOC_CMD, }` from the user client program, it sends the command to the next interface through the output `FPIPE` and signals this exceptional condition to the kernel, by sending an `USR_EOT` event (see 1.3.8). In this case, the kernel just waits for all the interfaces to stop.

Once all the interfaces are stopped, the turn pointers are changed, the user is signalled to start talking and all the corresponding interfaces are started.

### 1.2.2 End of call

An end of call is handled differently whether is the talking user who finishes the call, or the listening user. In the first case, the platform needs to wait until all the audio is been recognized, translated and synthesized to the listening user, before finishing the call. In the second case, when the listening user hangs up the phone, there is no need to wait for anything, and all the interfaces are aborted.

If the user who is talking hangs up the phone, the `UserInterface` assigned to the talking user receives an `{EOC_CMD, }` from the socket. It passes it to the next interface through the `FPIPE`, signals the kernel with an `EOC_TLK` and exits. Each interface in turn, when receiving the `{EOC_CMD, }`, will stop the corresponding server and exit. The kernel will wait for all the interfaces to be done, and will finish the call (see 1.3.1).

On the other hand, if the user who is listening finishes the call, the `UserInterface` receiving the `{EOC_CMD, }` will signal the kernel an `EOC_LST` event and will exit. The kernel will then abort all the interfaces and finish the call (see 1.3.7).

### 1.2.3 Errors

Each interface is prepared to handle some errors: network errors (the connection with the server gets broken) and internal server errors (the server sends an unexpected `{ABORTED_CMD, }` command). In both cases, the interface acts the same way: signals the kernel and exits. The event signalled depends on each particular platform.

The possible events are:

- `U_LTK_ABORT` Fatal error, generated by talking `UserInterface` (see section 1.3.3).
- `U_LST_ABORT` Fatal error, generated by listening `UserInterface` (see section 1.3.2).
- `ASR_ABORT` Fatal error, generated by `AsrInterface` (see section 1.3.4).
- `TTT_ABORT` Fatal error, generated by `TttInterface` (see section 1.3.6).
- `TTS_ABORT` Fatal error, generated by `TtsInterface` (see section 1.3.5).

## 1.3 Event procedures

### 1.3.1 EOC\_TLK

EOC_TLK	UI.TLK.wait() ASR.wait() TTT.wait() TTS.wait() UI.LST.wait()	Wait for interfaces to finish
	FP_UI.reset() FP_ASR.reset() FP_TTT.reset() FP_TTS.reset()	Reset internal pipes
	delete UI_1 delete ASR_1 delete TTT_1 delete TTS_1 delete UI_2 delete ASR_2 delete TTT_2 delete TTS_2	Reset internal pipes
	close & delete SOCK_UI_1 close & delete SOCK_ASR_1 close & delete SOCK_TTT_1 close & delete SOCK_TTS_1 close & delete SOCK_UI_2 close & delete SOCK_ASR_2 close & delete SOCK_TTT_2 close & delete SOCK_TTS_2	Close & delete sockets

### 1.3.2 ULLST\_ABORT

ABORT	UI_TLK.quit() UI_TLK.wait()	Signal interfaces to stop immediately
	FP_ASR.stop() ASR.quit() ASR.wait()	
	FP_TTT.stop() TTT.quit() TTT.wait()	
	FP_TTS.stop() TTS.quit() TTS.wait()	
	FP_UI.stop() ULLST.quit() ULLST.wait()	
FP_UI.reset() FP_ASR.reset() FP_TTT.reset() FP_TTS.reset()	Reset internal pipes	
delete UI_1 delete ASR_1 delete TTT_1 delete TTS_1 delete UI_2 delete ASR_2 delete TTT_2 delete TTS_2	Reset internal pipes	
close & delete SOCK_UI_1 close & delete SOCK_ASR_1 close & delete SOCK_TTT_1 close & delete SOCK_TTS_1 close & delete SOCK_UI_2 close & delete SOCK_ASR_2 close & delete SOCK_TTT_2 close & delete SOCK_TTS_2	Close & delete sockets	

### 1.3.3 UI\_TLK\_ABORT

Same as section 1.3.2.

### 1.3.4 ASR\_ABORT

Same as section 1.3.2.

### 1.3.5 TTS\_ABORT

Same as section 1.3.2.

### 1.3.6 TTT\_ABORT

Same as section 1.3.2.

### 1.3.7 EOC\_LST

EOC_LST	UI_TLK.quit() UI_TLK.wait()	Signal interfaces to stop immediately
	FP_ASR.stop() ASR.quit() ASR.wait()	
	FP_TTT.stop() TTT.quit() TTT.wait()	
	FP_TTS.stop() TTS.quit() TTS.wait()	
	FP_UI.stop() UI_LST.wait()	
FP_UI.reset() FP_ASR.reset() FP_TTT.reset() FP_TTS.reset()	Reset internal pipes	
delete UI_1 delete ASR_1 delete TTT_1 delete TTS_1 delete UI_2 delete ASR_2 delete TTT_2 delete TTS_2	Reset internal pipes	
close & delete SOCK_UI_1 close & delete SOCK_ASR_1 close & delete SOCK_TTT_1 close & delete SOCK_TTS_1 close & delete SOCK_UI_2 close & delete SOCK_ASR_2 close & delete SOCK_TTT_2 close & delete SOCK_TTS_2	Close & delete sockets	

EOC_LST	UI_TLK.quit() ASR.quit() TTT.quit() TTS.quit()	Signal interfaces to stop immediately
	UI_TLK.wait() ASR.wait() TTT.wait() TTS.wait() UILLST.wait()	Wait for interfaces to finish
	FP_UI.reset() FP_ASR.reset() FP_TTT.reset() FP_TTS.reset()	Reset internal pipes
	delete UL1 delete ASR_1 delete TTT_1 delete TTS_1 delete UL2 delete ASR_2 delete TTT_2 delete TTS_2	Reset internal pipes
	close & delete SOCK_UL1 close & delete SOCK_ASR_1 close & delete SOCK_TTT_1 close & delete SOCK_TTS_1 close & delete SOCK_UL2 close & delete SOCK_ASR_2 close & delete SOCK_TTT_2 close & delete SOCK_TTS_2	Close & delete sockets

### 1.3.8 ASR\_EOT and UI\_EOT

ASR_EOT	UI_TLK.stop()	Stop UserInterface
UI_EOT	UI_TLK.wait() ASR.wait() TTT.wait() TTS.wait() UI_LST.wait()	Wait for interfaces to finish
	FP_UI.reset() FP_ASR.reset() FP_TTT.reset() FP_TTS.reset()	Reset internal pipes
	Set pointers	Set pointers according to table 1.2
	FP_UI.start() FP_ASR.start() FP_TTT.start() FP_TTS.start()	Start internal pipes
	ASR.start() TTT.start() TTS.start() UI_LST.start() UI_TLK.start()	Start interfaces

Module	IP	Port
ASR_EN	147.83.50.14	1
ASR_ES	147.83.50.14	2
ASR_CA	147.83.50.14	3
TTT_EN	147.83.50.14	4
TTT_ES	147.83.50.14	5
TTT_CA	147.83.50.14	6
TTS_EN	147.83.50.14	7
TTS_ES	147.83.50.14	8
TTS_CA	147.83.50.14	9
DialogicUser (Server)	147.83.50.14	10

Table 1.1: Stored IP addresses

Pointer	User 1 talking	User 2 talking
UI.LST	UI.2	UI.1
UI.TLK	UI.1	UI.2
ASR	ASR.1	ASR.2
TTT	TTT.1	TTT.2
TTS	TTS.1	TTS.2
FP_UI	FP_UI.1	FP_UI.2
FP_ASR	FP_ASR.1	FP_ASR.2
FP_TTT	FP_TTT.1	FP_TTT.2
FP_TTS	FP_TTS.1	FP_TTS.2

Table 1.2: Pointers assignment.

## 1.4 CommonInterface

API functions	Actions	Answer
<code>stop()</code>	<pre> if state != STARTED, return; wait input thread to finish set state STOPPED send {STOP_CMD, -} wait output thread to finish set state STOPPED </pre>	<code>{STOPPED_CMD, -}</code>
<code>quit()</code>	<pre> if state != STARTED, return; set state INSTOP stop input FPIPE wait input thread to finish set state ABORTED send {ABORT_CMD, -} wait output thread to finish set state STOPPED </pre>	<code>{ABORTED_CMD, -}</code>
<code>wait()</code>	<pre> if state != IDLE, return; wait input thread to finish wait output thread to finish set state IDLE </pre>	

## 1.5 UserInterface

The *UserInterface* module implements the interface with the different type of users: Dialogic, Microphone and Console. It provides some functions to start/stop it, to quit it, to set the turn of the user and to wait until it is finished. During normal operation, the *UserInterface* reads commands from the input FPIPE, performs the associated actions (sending/reading command to/from the socket connected to the user), and writes commands to the output FPIPE.

### 1.5.1 UserInterface. Input thread

The thread reads command packets from a FPIPE, and sends command packets through the socket.

Command	Actions	Answer
{EOC_CMD, -}	set state=EOC send {STOP_CMD, -} pthread_exit()	{STOPPED_CMD, -}
{EOT_CMD, -}	set state=EOT send {STOP_CMD, -} pthread_exit()	{STOPPED_CMD, -}
{AUDIO_DATA, length}	send {AUDIO_DATA, length} send length bytes of data	

### 1.5.2 UserInterface. Output thread

The thread reads command packets from the socket, and sends command packets to the FPIPE.

Command	Actions
{STOPPED.CMD, -}	set state=STOPPED pthread_exit()
{ABORTED.CMD, -}	if state!=ABORTED, send event UI_ABORTED set state=STOPPED pthread_exit()
{EOC.CMD, -}	if (turn) send command {EOC.CMD, -} send event EOC_TLK else send event EOC_LST set state=STOPPED pthread_exit()
{EOT.CMD, -}	sent event UI_EOT send command {EOT.CMD, -} set state=STOPPED pthread_exit()
{AUDIO.DATA, length}	send {AUDIO.DATA, length} send <i>length</i> bytes of data

## 1.6 AsrInterface

### 1.6.1 AsrInterface. Input thread

The thread reads command packets from a FPIPE, and sends command packets through the socket.

Command	Actions	Answer
{EOC_CMD, -}	set state=EOC send {STOP_CMD, -} pthread_exit()	{STOPPED_CMD, -}
{EOT_CMD, -}	set state=EOT send {STOP_CMD, -} pthread_exit()	{STOPPED_CMD, -}
{AUDIO_DATA, length}	send {AUDIO_DATA, length} send <i>length</i> bytes of data	

### 1.6.2 AsrInterface. Output thread

The thread reads command packets from the socket, and sends command packets to the FPIPE.

Command	Actions
{STOPPED_CMD, -}	if state==EOC, send {EOC_CMD, -} else if state==EOT, send {EOT_CMD, -} set state=STOPPED set input FPIPE RD_NOWAIT pthread_exit()
{ABORTED_CMD, -}	if state!=ABORTED, send event ASR_ABORTED set state=STOPPED pthread_exit()
{EOT_CMD, -}	send event ASR_EOT send command {EOT_CMD, -} set state=STOPPED pthread_exit()
{RECINFO_CMD, index}	if first_struct, allocate memory read sizeof(RecInfo) bytes from socket if begin > end, discard_text = true
{TEXT_DATA, length}	read length bytes of data if discard_text == false, send {TEXT_DATA, length} + text data if last struct, write recognition results

The whole bunch of recognized text is divided into several recognized units, each of them associated with a RecInfo struct. These structs contain information about time alignment and recognition probability.

The protocol to receive the recognized text from the server is as follows: Before sending each recognized unit, the corresponding RecInfo struct is sent with a {RECINFO\_CMD, index}. The *index* of the first {RECINFO\_CMD, index} indicates how many structs are going to be sent (including that). The index decreases with each {RECINFO\_CMD, index} received (the index equals 1 when the last struct is received). Keep in mind that if the first RecInfo command received indicates that there are  $n$  structs in the set, we need to save space for  $n+1$  structs, since the last one must be NULL.

Each recognized word is assigned to a RecInfo struct. Before receiving a {TEXT\_DATA, }, a {RECINFO\_CMD, index} is sent. To show the results, each word must be stored in the char pointer in the corresponding RecInfo struct.

In case of a begin index greater than the end index, the text is going to be received does not contain anything useful, so the *discard\_text* flag is activated. If this flag is NOT activated, when the {TEXT\_DATA, } is received, the text is also sent to the input FPIPE.

## 1.7 TttInterface

### 1.7.1 TttInterface. Input thread

The thread reads command packets from a FPIPE, and sends command packets through the socket.

Command	Actions	Answer
{ <b>EOC_CMD</b> , -}	set state=EOC send { <b>STOP_CMD</b> , -} pthread_exit()	{ <b>STOPPED_CMD</b> , -}
{ <b>EOT_CMD</b> , -}	set state=EOT send { <b>STOP_CMD</b> , -} pthread_exit()	{ <b>STOPPED_CMD</b> , -}
{ <b>TEXT_DATA</b> , <i>length</i> }	send { <b>TEXT_DATA</b> , <i>length</i> } send <i>length</i> bytes of data	

### 1.7.2 TttInterface. Output thread

The thread reads command packets from the socket, and sends command packets to the FPIPE.

Command	Actions
{ <b>STOPPED_CMD</b> , -}	if state==EOC, send { <b>EOC_CMD</b> , -} else if state==EOT, send { <b>EOT_CMD</b> , -} set state=STOPPED pthread_exit()
{ <b>ABORTED_CMD</b> , -}	if state!=ABORTED, send event TTT_ABORTED set state=STOPPED pthread_exit()
{ <b>TEXT_DATA</b> , <i>length</i> }	send { <b>TEXT_DATA</b> , <i>length</i> } send <i>length</i> bytes of data

## 1.8 TtsInterface

### 1.8.1 TtsInterface. Input thread

The thread reads command packets from a FPIPE, and sends command packets through the socket.

Command	Actions	Answer
{EOC_CMD, -}	set state=EOC send {STOP_CMD, -} pthread_exit()	{STOPPED_CMD, -}
{EOT_CMD, -}	set state=EOT send {STOP_CMD, -} pthread_exit()	{STOPPED_CMD, -}
{TEXT_DATA, length}	send {TEXT_DATA, length} send <i>length</i> bytes of data	

### 1.8.2 TtsInterface. Output thread

The thread reads command packets from the socket, and sends command packets to the FPIPE.

Command	Actions
{STOPPED_CMD, -}	if state==EOC, send {EOC_CMD, -} else if state==EOT, send {EOT_CMD, -} set state=STOPPED pthread_exit()
{ABORTED_CMD, -}	if state!=ABORTED, send event TTS_ABORTED set state=STOPPED pthread_exit()
{AUDIO_DATA, length}	send {AUDIO_DATA, length} send <i>length</i> bytes of data

## Chapter 2

# Platform users

### 2.1 Microphone

#### 2.1.1 Initial

Send to platform	Receive from platform
{ <b>IFACE_CMD</b> , <i>MICROPHONE</i> }	
{ <b>CFG_CMD</b> , <i>integer code (see fig. 2.1.2, p. 25)</i> }	{ <b>LANG_CMD</b> , [ <i>language selected</i> ]}
If client selected an IP address:	
{ <b>IP_CMD</b> , [ <i>ip address length</i> ]} IP address as (char *) { <b>PORT_CMD</b> , [ <i>port</i> ]}	{ <b>IP_CMD</b> , —}
If client selected a PHONE number:	
{ <b>PHONE_CMD</b> , [ <i>phone number length</i> ]} Phone number (char *)	{ <b>PHONE_CMD</b> , —}
If client selected other:	
Send to platform NOTHING	Receive from platform

code[0]	Language origin
1	English
2	Spanish
3	Catalan
code[1]	Language destination
1	English
2	Spanish
3	Catalan
0	User selects
code[2]	Destination type
0	IP address
1	Phone number
2	Loop
3	None (Audio file)
4	None (Text file)
code[3-]	Configuration number
0	Selects <i>config0.txt</i>
1	Selects <i>config1.txt</i>
:	

Figure 2.1: Fields of the integer code

### 2.1.2 Normal

Send to platform	Receive from platform
{TURN_CMD, [integer: 1 or 0]} if turn is 1: send data otherwise: receive data {START_CMD, -}	{STARTED_CMD, -}
If turn 1:	
If turn 0:	

## 2.2 DialogicUser

Dialogic card user interface. Provides telephone access to the translation platform. It can run in two different modes:

- Client mode: waits for an incoming call, configures the user, and connects with the translation platform.

- Server mode: dial—out program when requested by the translation platform.

### 2.2.1 Client mode

User Interface	Wait for user call Ask for language Ask for destination type (IP or PHONE) if PHONE ask for phone number else if IP ask for IP address ask for port number
Platform configuration	Connect socket to server Send { <b>IFACE_CMD</b> , <i>DIALOGIC</i> } Send { <b>LANG_CMD</b> , <i>language</i> } if PHONE Send { <b>PHONE_CMD</b> , <i>length</i> } Send data: phone number else if IP send { <b>IP_CMD</b> , <i>length</i> } send data: ip address send { <b>PORT_CMD</b> , <i>port number</i> }
Normal interface	(See below)

### 2.2.2 Server mode

Server initialization	Receive { <b>PHONE_CMD</b> , <i>phone length</i> } Receive data: phone number
User configuration	Dial phone number Ask user language
Platform configuration	Send { <b>IFACE_CMD</b> , <i>DIALOGIC</i> } Send { <b>LANG_CMD</b> , <i>language</i> }
Platform normal interface	See below

### 2.2.3 Dialogic thread

TDX_CST->DE_TONEON	<i>User hanged up</i>  if (datap->cst_data==DISCTONE) send {EOC_CMD, -} set state=STOPPED exit thread
TDX_PLAY->TM_USRSTOP	<i>Channel stopped while playing</i>  set state=STOPPED exit thread
TDX_PLAY->TM_EOD	<i>End of data reached while playing</i>  set state=STOPPED exit thread
TEC_STREAM	<i>End of data reached while streaming</i>  set state=STOPPED exit thread

### 2.2.4 Input/ouput thread

The DialogicUser is controlled by the Platform through a socket, sending special command packets. The DialogicUser program communicates with the platform also sending command packets. The tables below show the different commands.

## Input commands

{START_CMD, —}	<i>Start the streaming/playing of audio to/from application.</i> Start Dialogic thread Wait until state==STARTED Send {STARTED_CMD, —}
{STOP_CMD, —}	<i>Stop the streaming/playing of audio to/from application when all available data has been used.</i> Signal internal FPIPE to empty the buffer and exit Wait until state==STOPPED Send {STOPPED_CMD, —} Reset FPIPE
{ABORT_CMD, —}	<i>Abort the streaming/playing of audio to/from application immediately.</i> Stop DialogicController asynchronously Wait until state==STOPPED Send {ABORTED_CMD, —} Reset FPIPE
{TURN_CMD, new_turn}	Set the new turn of the user Set DialogicController turn to new_turn
{AUDIO_DATA, length}	Audio data to play to user Write length bytes of data from socket into FPIPE

## Output commands

{STARTED_CMD, —}	The streaming/playing of audio has been started
{STOPPED_CMD, —}	The streaming/playing of audio has finished
{ABORTED_CMD, —}	The streaming/playing of audio has been aborted
{EOC_CMD, —}	The call has been terminated by the user hanging up the phone
{AUDIO_DATA, length}	Audio data recorder from user

## Playing/streaming audio data

During the normal conversation flow, when the user is *talking*, the Dialogic card is streaming data to the application (*DialogicUser*), and when the user is *listening*, the Dialogic card is playing data to the user.

**Data streaming.** The Dialogic card fills an internal buffer with the recorded audio data. When the buffer is full, a callback function is called with the address of the buffer and the amount of data in it. The application sends a data packet `{AUDIO_DATA, length}` to the Platform through the socket, and then the amount of data indicated before. Streaming stops when the Dialogic card is stopped (with a `{STOP_CMD, -}` command) or aborted (with a `{ABORT_CMD, -}` command or due to an unexpected error).

**Data playing.** The DialogicUser receives a data packet `{AUDIO_DATA, length}` indicating the amount of audio data is going to be sent through the socket. The DialogicUser reads `length` bytes of audio data from the socket, and writes them into the internal *FPIPE*. When the Dialogic card is playing, a modified version of the **write** function is used to write new data into the output buffers of the card. This modified function reads data from the internal *FPIPE* and writes it into the address of the output audio buffers. Playing stops when the *FPIPE* is empty and it is not being filled up with new data, or when the DialogicController is specifically stopped (due to an `{ABORT_CMD, -}` command received or an unexpected error).

**Hang up** When the user hangs up, both when talking or listening, the DialogicUser sends a `{EOC_CMD, -}` command to the platform and stops, getting ready for the next call.

### Starting DialogicUser

- *DialogicUser* receives `{START_CMD, -}` (sent by *TranslationPlatform*).
- It starts the thread controlling the Dialogic card and waits for the state to be changed to *STARTED*.
- The Dialogic thread starts the recording/playing process and sets the state to *STARTED*.
- The *DialogicUser* input/output module detects this change of state, and sends `{STARTED_CMD, -}` through the socket (to the *TranslationPlatform*).

### Stopping DialogicUser

- Platform sends command `{STOP_CMD, -}` through *ChannelInterface*.
- DialogicUser receives it and signals its internal *FPIPE* to empty the buffer and exit.
- DialogicControllers detects the *End Of Data* and stops, setting the state to **STOPPED**.

- DialogicUser detects this change of state, and sends command {**STOPPED\_CMD**, —} to the Platform.
- Platform receives this command confirming the channel has been stopped.

#### **Aborting DialogicUser**

- Platform sends command {**ABORT\_CMD**, —} through *ChannelInterface*.
- DialogicUser receives it and asynchronously stops the DialogicController and waits state to be **STOPPED**.
- DialogicUser resets its internal *FPIPE* and sends {**ABORTED\_CMD**, —} command to Platform.
- Platform receives this command confirming the channel has been aborted.

#### **Changing the turn**

- Platform sends command {**TURN\_CMD**, *turn*} through *ChannelInterface*.
- DialogicUser receives it and sets the new turn into the *DialogicController* (it takes effect next time the controller is started).

## Chapter 3

# GNU Free Documentation License

Version 1.2, November 2002

Copyright ©2000,2001,2002 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms

of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **”Document”**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **”you”**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **”Modified Version”** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **”Secondary Section”** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **”Invariant Sections”** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **”Cover Texts”** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **”Transparent”** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not **”Transparent”** is called **”Opaque”**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output

purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## **4. MODIFICATIONS**

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## **5. COMBINING DOCUMENTS**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## **6. COLLECTIONS OF DOCUMENTS**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into

the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## **7. AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## **8. TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## **9. TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## **10. FUTURE REVISIONS OF THIS LICENSE**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

### **ADDENDUM: How to use this License for your documents**

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.