The WireLess TelNet
voiXtreme voice system

Versions: VoiXtreme V250 / WireLess TelNet V440

Table of contents

1 Description

The TelNet voice system allows the PDA to say texts (TTS) and to perform ASR inputs using a “standard” telnet application (5250).

2 TTS system

Text data to be said (by TTS), may be supplied by:

- “Voice Triggers” containing constant and variable texts
- Error messages

3 ASR system

Voice input from user. The ASR system recognizes the user's voice and converts it into text data using “grammars”. Data input is processed by the local application, and then sent to host following programming configuration.

Grammars are recognition algorithms designed to produce text inputs.
Available grammars are:

- “Digits”: accepts decimal digits from “0” to “9”. More than one digit numbers are composed by repetition of digits. Digits should be preceded by a prefix and followed by a suffix to improve recognition.
- "Controls": accepts “Accept” and “Cancel” orders. *Accept* and, *Cancel* keywords are configurable and processed by the voice library, and host transmission is performed following input algorithm.
- “Functions”: accepts up to 20 keywords, which should be associated with a function key in the configuration file.
- “Custom”: 10 custom grammars allowing to independent activation of groups of keywords.

Keywords change according to the language selected for the ASR.

Each time a word is recognized by the ASR engine, a “Reliability Level” is returned indicating the score reached by the voice locution to match one of the words in grammar.
A high score (near 10.000) indicates that the ASR engine considers the user locution corresponds to the returned word with high level of probability. A low score indicates a low level of probability, so there is a high risk of misheard. It is recommended, in this case, to reject the recognition and wait for user input again. The returned reliability level changes following application environment, so it is possible to configure this option in configuration file.

ASR engine processes user locutions in three steps:

- Waiting for audio signal
- Capturing
- Recognition

In the first state the system waits for some audio signal. If this signal is significant, it considers that the user is speaking and starts the streaming capture. When audio signal is no longer detected, or a timeout is reached, the audio captured frames are processed by the recognizer. A configurable option (AsrThreshold) allows tuning the audio level to switch from one state to another.

When activated, the voice system will be in play mode (recognizing defined input words) or in pause mode (recognizing only resume words).

Predefined actions

Some keywords are processed internally by the voice library to perform some usual functions:

- Repeat command, to repeat to user the last voice announcement.
- Information command, to say to user useful information about the current trigger.
- Volume up/down command, to set up / down the TTS volume.
- Speed up/down command, to set up/down the TTS speed
- Speech-pause command, allowing to pause the ASR recognition engine to avoid invalid input.

These actions are performed internally. It is just needed to fill some configurable fields to adapt to current application.

User and noise calibration

The ASR engine automatically adapts the recognition to the current user and noise environment. It is recommended to reset the terminal to change the current user adaptation. An ASR calibration procedure is run at emulator start-up, allowing the user to say some words to calibrate the ASR engine to the current user and current environment noise. A configuration file allows setting critical words to perform calibration and validation of good recognition before working with the system.

4 Voice Triggers

Voice triggers are defined in three sections:

- *Conditions*: match needed to perform an event
- *TTS*: allows to give the user a voice instruction
- *ASR*: allows to get data from user

One or more triggers could be defined to perform voice events. Voice events can be TTS only, ASR only, and TTS + ASR.

4.1 Conditions

"Voice triggers" will start the voice input process. Their execution is determined by the presence of specific conditions on the emulated screen, such as:

- Text found / not found at a specific place of the screen
- Cursor position
- It is possible to match two screen areas

Up to five conditions about screen display may be defined.

4.2 TTS

In the Voice Trigger, a TTS "voice announcement" may be defined to be said before the ASR recognition process starts. This announcement may be a mix of constant text (defined inside the configuration file) and variable text (read from the emulated screen ("RCL") or contained inside a variable or result of a calculation). "RCL" text will be defined by beginning position and length: three numeric values separated by commas (row and column position in screen and text length) delimited by curly brackets: {r, c, l}.For example, {10, 7, 14} defines a text located at row 10, column 7, with 14 characters length. Variables are defined using "#N" where "N" is either a variable number of a symbol representing a special variable such as "$" (last user voice raw input) or "&" (last user voice formatted input). See chapter 7.2.4 for details about expressions.

Some processing may be done on text to say.

- Say or spell (<SPL> <SAY>)
- Uppercase or lowercase (U, L)
- Alpha or numeric trim (T, N)
- Suppress leftmost or rightmost characters (L, R)
- Take only leftmost or rightmost characters (l, r).

Up to three independent sentences are available. It is possible to conditionally say or not these sentences following the precedent trigger executed.

4.3 ASR

For the ASR recognition, you should define the ASR grammar to be used, and the ASR algorithm.

Standard voice grammars, "Digits", "Controls" and “Functions”, are included with the voiXtreme libraries. Custom grammars can be added. The grammars are activated independently in each trigger.

The ASR algorithm could be:

- *Single* (Confirm=No): The engine recognizes the user input, then sends it to host.
- *Single with repetition* (Repeat=Yes): The engine recognizes the user input, repeats it to user by TTS, then sends it to host.
- *Validate* (Confirm=Yes): The engine recognizes the user input, repeats it, and waits for a command said by user (“*accept*” / “*cancel*”):
  - If *cancel*, the application restarts the trigger from the beginning (voice announcement).
  - If *accept*, the application sends user input to host.
- *Validate with repetition* (Repeat=Yes): Same as Validate, but the validation command is echoed to user.

See below the different input modes.

Some options allow to validate user input, such as min and max digits.

A format option allows to reformat output strings to combine user input (represented by {$} or {#$}), and constants or variable data (screen text, variables, calculations).

A check option allows to mandatory match user input with constant or variable data to allow data output. It will be useful to validate input in transactions when the user response must match some known value (i/e location, product code, etc).

Informational options allow helping the user in a working flow to know what is expected to say.

5 Input Modes

The following images describe the different input modes.

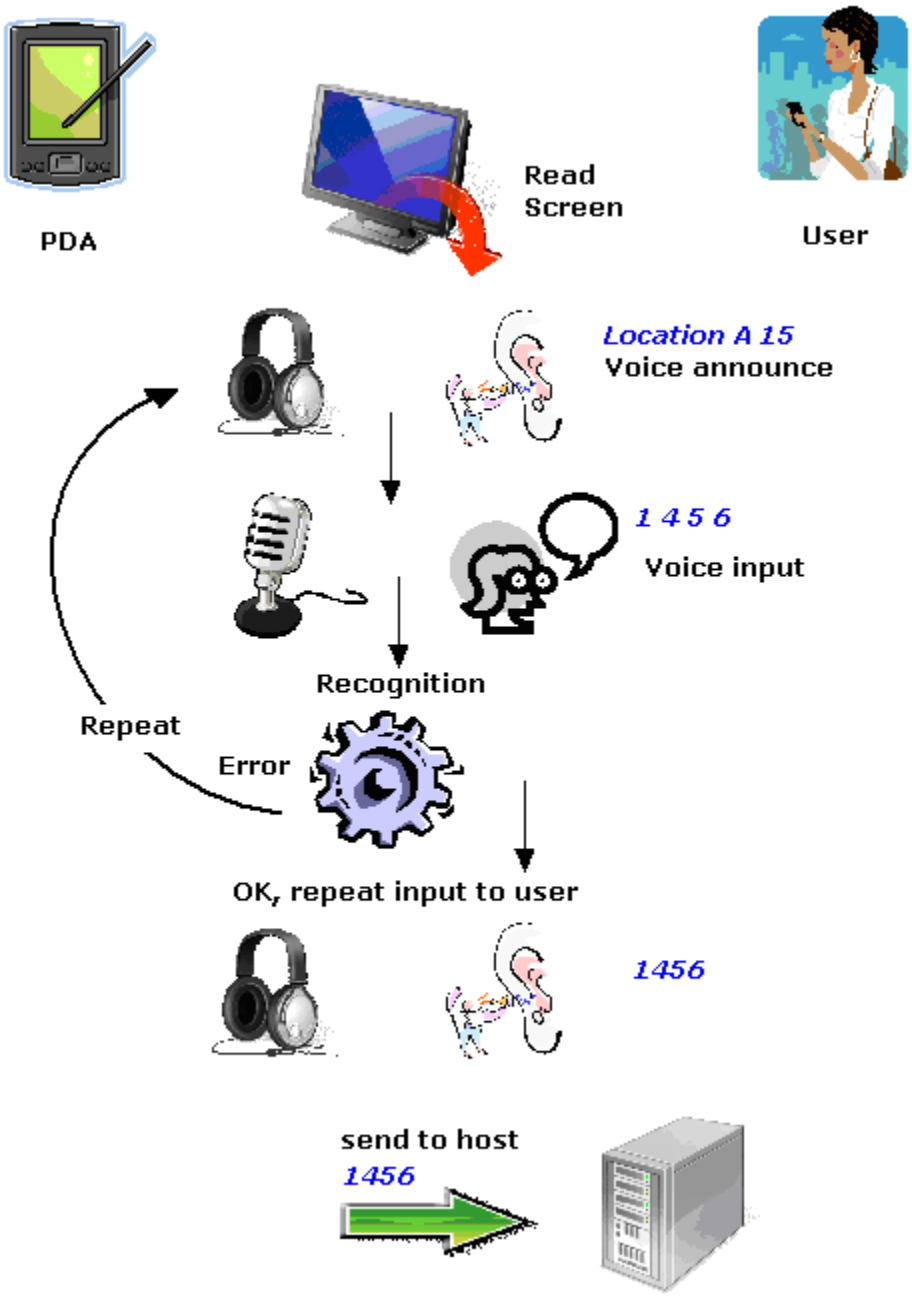
5.1 Single Voice Input

5.1.1 Single Voice Input Algorithm:

1/ A defined trigger matches screen data and cursor position.
2/ The TTS section of the trigger is processed to say the voice announcement to the user.
3/ Waits the end of TTS announcement.
4/ The ASR section of trigger is processed to perform the ASR input, using the grammar indicated in the section (on this case, "Digits").
5/ End of recognition:

- *Repeat*, goes to (2).
- Error (no recognition, low reliability level, min / max) goes to (2).
- *Cancel*, sends the [VOICETRIGGER_ASR_XX]/Cancel string to Host, then exit.
- *Digits*, sends the input followed by [VOICETRIGGER_ASR_XX]/Terminator string to Host, then exit.

5.2 Simple + Repeat Voice Input

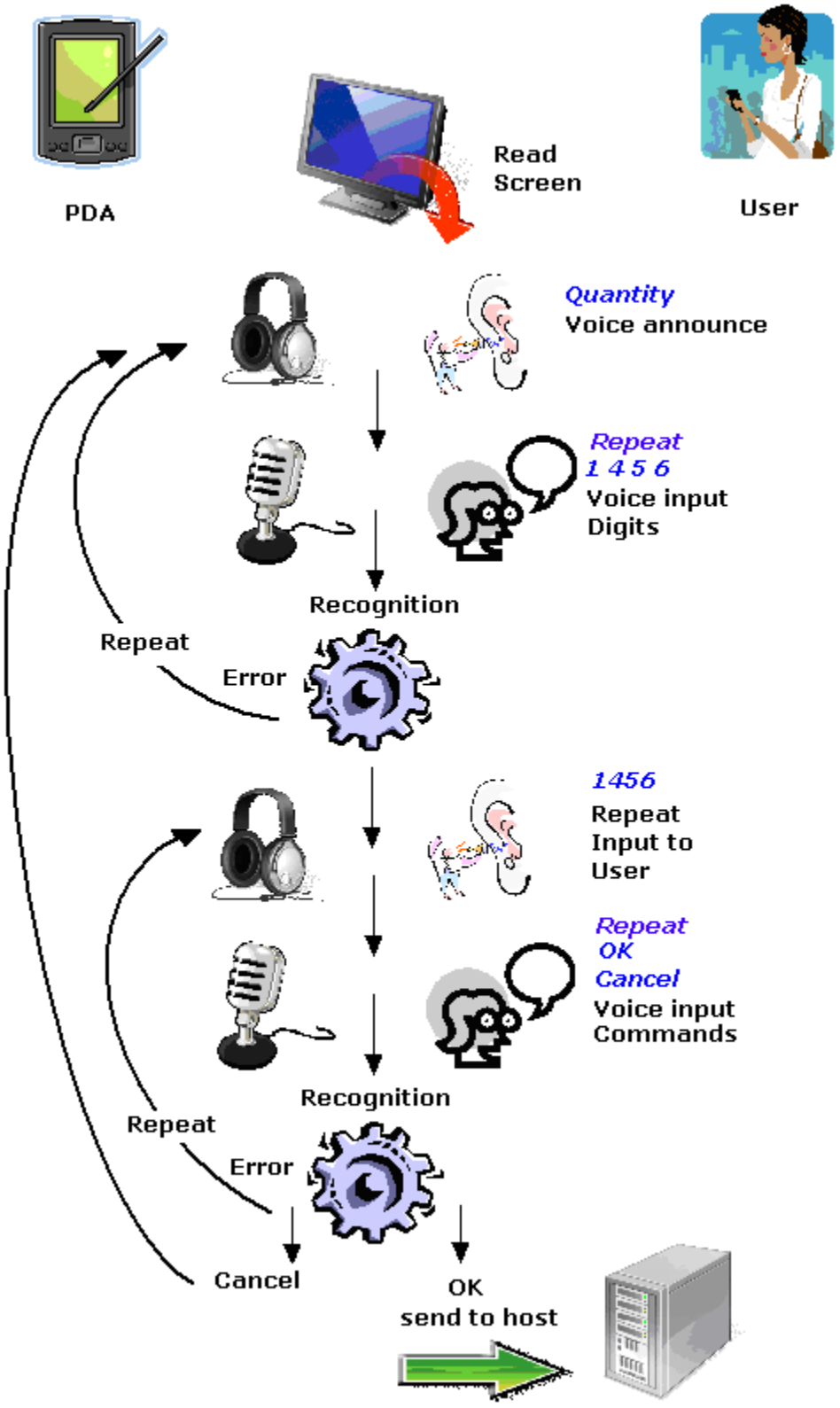


5.2.1 Simple + Repeat Voice input algorithm:

1/ A defined trigger matches screen data and cursor position.
2/ The TTS section of the trigger is processed to say the voice announcement to user.
3/ Waits the end of TTS announcement.
4/ The ASR section of the trigger is processed to perform the ASR input, using the grammar indicated in the section (on this case, "Digits").
5/ End of recognition:

- *Repeat*, goes to (2).
- Error (no recognition, low reliability level, min / max) goes to (2).
- *Cancel*, says "Cancel" to user, sends the [VOICETRIGGER_ASR_XX]/Cancel string to Host, then exit.
- *Digits*, says the input to user, sends the input followed by [VOICETRIGGER_ASR_XX]/Terminator string to Host, then exit.

5.3 Validate Voice Input (input digits)



5.3.1 Validate Voice input algorithm (digits):

1/ A defined trigger matches screen data and cursor position.
2/ The TTS section of the trigger is processed to say the voice announcement to user.
3/ Waits the end of TTS announcement.
4/ The ASR section of the trigger is processed to perform the ASR input, using the grammar indicated in the section (on this case, "Digits").
5/ End of recognition:

- Repeat, goes to (2).
- Error (no recognition, low reliability level, min / max) goes to (2).
- Cancel, goes to (9).
- Digits, goes to (6).

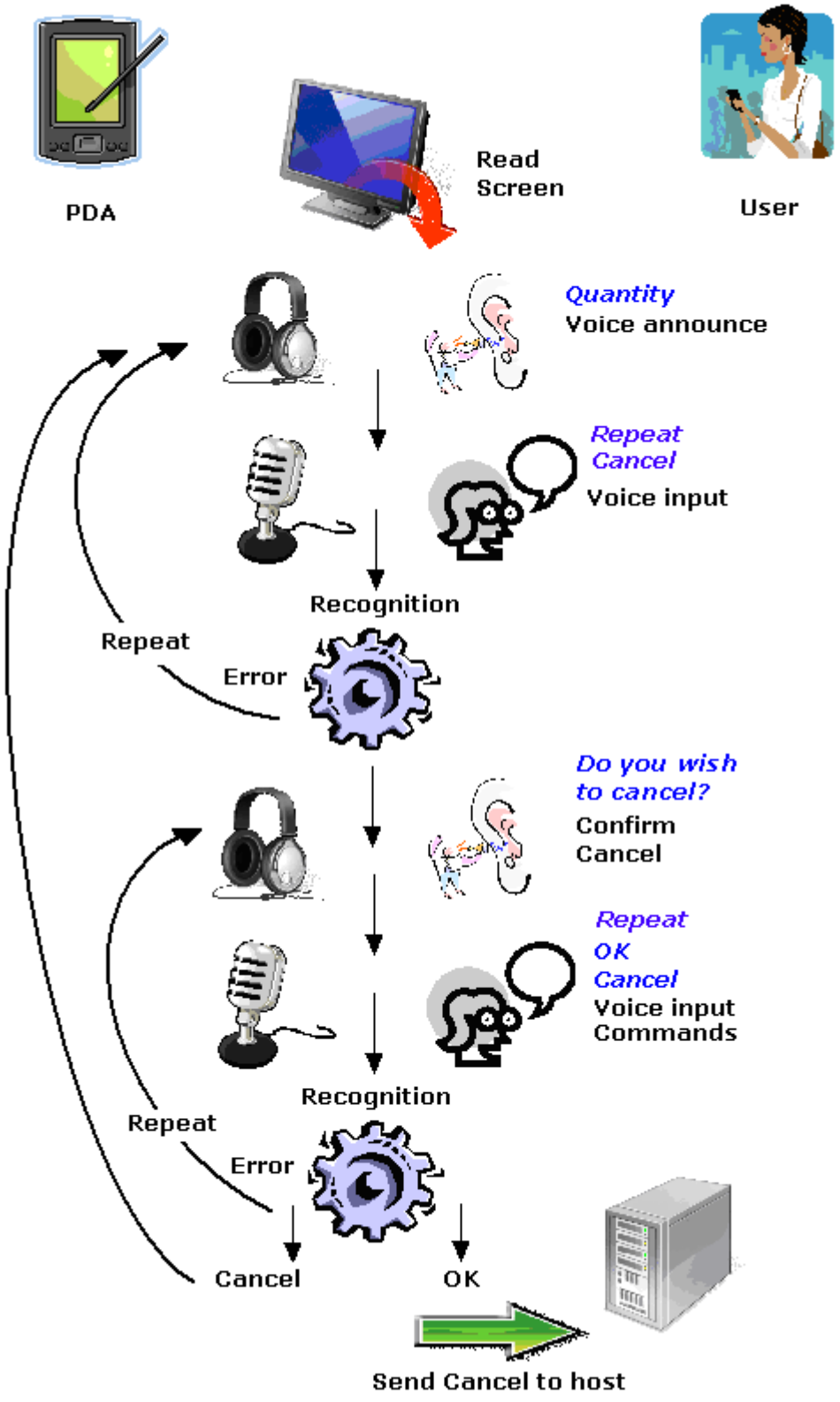
6/ Says the input to user by TTS.
7/ Performs the ASR recognition, using the “Commands” grammar.
8/ End of recognition:

- Repeat, goes to (6).
- Error (no recognition, low reliability level, min / max) goes to (6).
- Cancel, goes to (6).
- OK, says "OK" to user, sends the input followed by [VOICETRIGGER_ASR_XX]/Terminator string to Host, then exit.

9/ Says the CancelConfirmation phrase to user by TTS.
10/ Perform the ASR recognition, using the “Commands” grammar.
11/ End of recognition:

- Repeat, goes to (9).
- Error (no recognition, low reliability level, min / max) goes to (9).
- Cancel, says "Cancel" to user, goes to (2).
- OK, says "OK" to user, sends the [VOICETRIGGER_ASR_XX]/Cancel string to Host, then exits.

5.4 Validate Voice Input (input cancel)



5.4.1 Validate Voice input algorithm (cancel):

1/ A defined trigger matches screen data and cursor position.
2/ The TTS section of the trigger is processed to say the voice announcement to user.
3/ Waits the end of TTS announcement.
4/ The ASR section of the trigger is processed to perform the ASR input, using the grammar indicated in the section (on this case, "Digits").
5/ End of recognition:

- Repeat, goes to (2).
- Error (no recognition, low reliability level, min / max) goes to (2).
- Cancel, goes to (9).
- Digits, goes to (6).

6/ Says the input to user by TTS.
7/ Performs the ASR recognition, using the "Commands" grammar.
8/ End of recognition:

- Repeat, goes to (6).
- Error (no recognition, low reliability level, min / max) goes to (6).
- Cancel, goes to (6).
- OK, says "OK" to user, sends the input followed by [VOICETRIGGER_ASR_XX]/Terminator string to Host, then exit.

9/ Says the CancelConfirmation phrase to user by TTS.
10/ Performs the ASR recognition, using the "Commands" grammar..
11/ End of recognition:

- *Repeat*, goes to (9).
- Error (no recognition, low reliability level, min / max) goes to (9).
- *Cancel*, says "Cancel" to user, goes to (2).
- *OK*, says "OK" to user, sends the [VOICETRIGGER_ASR_XX]/Cancel string to Host, then exits.

6 TTS Only modes

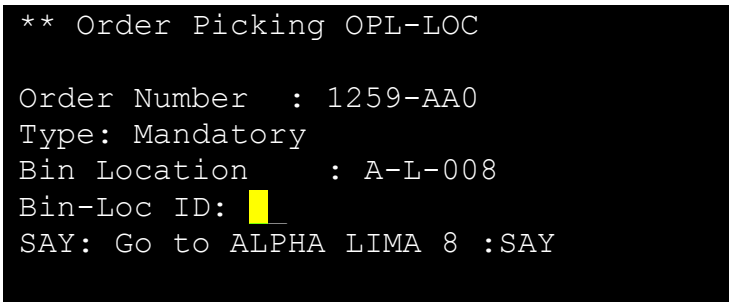
These modes are intended to be used in a “TTS” only application. Users will hear the terminal, but input data by scanner or keyboard (no ASR recognition).

Do not use these modes in the same application with voice triggers.

6.1 Read screen

Read Screen sample

In this sample, a part of the screen (delimited by SAY: / :SAY) is read to user by TTS.



The text delimited by the tags “SAY: - :SAY” is read out loud to user as: *“go to alpha lima eight*”.

7 Configuration File

Here below there's the description of the Voice configuration file.

The configuration file follows a “dot-ini” format with sections (delimited by brackets), keys (keywords inside the section, next to "=" sign) and values (text next to "=" sign). Sections and keys are in special font: **SECTION**. Values are in green.

7.1 Global definitions about the voice engine

7.1.1 Global TTS (Text-to-speech) section

[TTS_DEF]

This section groups the global Text-To-Speech (TTS) settings, and enables to process the screen and the printer for voice actions.

TtsPrinterOn= No
Not used in 5250 mode.

TtsErrorOn= Yes
Allows TTS engine to read the error message displayed to the user.
Validation error occurs automatically after reading.

7.1.1.1 User messages

Messages that will be said to user under specific conditions.

MsgWelcome= Welcome to Telnet by soff to go.
Message to say at start-up.

MsgConnecting= Connecting.
Message to say when connecting.

MsgConnected= Connected.
Message to say when TelNet connects to host.

MsgDisconnect= End of connection
Message to say when TelNet’s connection ends.

7.1.1.2 Keys to control voice engine by user

KeyRepeat= 0000
Key to repeat last text speech.

KeyVolUp= 0923
KeyVolDn= 0921
Keys to change the volume.

KeySpeedUp= 0922
KeySpeedDn= 0920
Keys to change the TTS speed.

7.1.2 Global ASR (Automatic Speech Recognition) definitions section

[ASR_DEF_TUNNING]
This section groups the ASR engine settings to tune the ASR engine.

AsrOn= Yes
Enable or disable the ASR engine. Set to No for TTS only applications.

ReliabilityLvl= 4500
Minimum reliability level (for digits) necessary to accept a word recognition. When a word is recognized, ASR engine returns a "reliability level" which is the confidence level that the engine assigns to the recognition. A low reliability level denotes a confusing recognition that will be rejected.

ReliabilityCmd= 4500
Minimum reliability level for commands (functions and controls) necessary to accept a keyword recognition (controls or functions).

ReliabilityExp= 300
Minimum reliability level for expected data (see expected in trigger ASR section) necessary to accept a word recognition.

7.1.2.1 Custom grammars

[ASR_DEF_GRAMMAR]
This section groups the names of the customized grammars (up to 10).
The grammars names are not significant for application, that just must match the name of the grammar in BNF file. It allows to independently activate groups of words (usually

keywords) depending on application requirements. It is useful to activate in one trigger only the keywords that are useful in the specific transaction.

CustomGrammar0=< custom0>
Name of the grammar referenced by the input mode parameter VOICETRIGGER_ASR / Grammars=0

CustomGrammar1=<custom1>
Name of the grammar referenced by the input mode parameter VOICETRIGGER_ASR / Grammars=1
...

CustomGrammar9=<custom9>
Name of the grammar referenced by the input mode parameter VOICETRIGGER_ASR / Grammars=9

7.1.2.2 Keywords

[ASR_DEF_KEYWORDS]
This section groups the usage of some keywords. The keywords must be defined in the grammar to be recognized. Upon a keyword is recognized, the action performed by this keyword is defined in this table.
These words must be configured in this file, and also be part of the ASR grammar (see file " `cd_en.bnf`" normally "`\SofToGo\WireLessTelNet\WTnCE52\Voice\EN`").
If a keyword is defined in the grammar but is not defined in this table it will be consider as data and sent “as is” as user input.
If a keyword is defined in this section, but is not present in the grammar it will never be recognized.
To be recognized, a keyword must be in an active grammar.

ResAccept=Ok
This is the keyword of the "controls" grammar the user should say to accept an action.

ResCancel=Cancel
This is the keyword of the "controls" grammar the user should say to cancel an action.

ResCalcEnd=missing
This is a keyword used in “Calculator” mode. It forces the calculator input mode to terminate even if the requested quantity is not reached. The current quantity ant the current terminator is sent to the host application. This keyword must be present in the <calculator> grammar.

ResCalcInfo=already-prepared
This is a keyword used in “Calculator” mode. It performs a informational customizable announce (CalcInfo) about the total expected and remaining quantity. This keyword must be present in the <calculator> grammar.

KeywordFn01=January|first
This keyword of the “functions” grammar, allows to generate the F1 function key. The sequence sent to host depends of the terminal emulation selected.
It is possible to set one or more keywords (separated by a “pipe”) to generate a function key. In case of multiple words for one key, it is recommended to set in different grammars to activate them separately. This feature is useful to handle different actions on the same key depending on context. I/e in one trigger the F1 key means to skip a picking location, in other trigger it will mean to cancel an order, in this case it is possible to set “skip|cancel” and activate one of them depending on context.
Up to 20 keywords can be set to perform from F1 to F20 function keys.

7.1.2.3 Keyboard keys by keywords

[ASR_DEF_KBD_KEYWORDS]
This section groups the keywords that will perform some specific keyboard actions (other than function keys) needed on application by a recognized keyword.

When the keyword is recognized, the associated keyboard scan-code is sent to keyboard processor. This feature allows to perform actions intended to be performed by user on keyboard (i/e ctrl-x to close the telnet session).

Scc00=0018
The scan-code 00 to be generated when the associated keyword is recognized.

Kwd00=disconnect
The keyword to generate the associated scan-code. It is mandatory needed to this keyword to be present in any grammar to be recognized.

There are some specific meta-codes (FFF0 – FFFF):

Say a variable content:
Scc00=FFF0 to Scc00=FFF9
the code FFF0 will say the content of the variable number {#0}
the code FFF9 will say the content of the variable number {#9}

Perform a special formatting action in a trigger:
The code Scc00=FFFF is the format keyword (see the FormatKw key in the ASR section) for the current trigger.

7.1.2.4 ASR engine calibration

[ASR_DEF_CALIBRATE]
This section groups the options for the ASR calibrate procedure. The Asr calibrate allows the ASR engine to adapt to current user and noise environment conditions.

The data calibrate file (AsrCalibrate.txt) is a configurable file containing a list of testing words to show/say to user and recognize.
This file has any number of text lines (separated by CR), and each line has four fields separated by semicolon (;).

The fields of each line are:
<type>;<Result>;<Display text>;<Voice announce>
* The Type is C for commands and D for digits
* The 'result' string to obtain (must be exactly the same than the grammars)
* The 'display text' to be shown in the dialog box.
* The 'voice announce' to be said to the user.

TargetReliability=5000
This is the target reliability level to apply to validate a recognized word.

TargetOk=5
This is the number of validated words the user is challenged to perform to terminate the calibrate procedure.
A valid word is a good response + the reliability level greater than **TargetReliability**.

SequenceMode=0
The mode to play the customized calibrate file.
0=True random mode with repetition
1=Random mode without repeating the same word
2=Sequential
3= By user selection (combo box)

SkipOnEr=Yes
Yes= If the word is not recognized, goto next word.
No= Repeat current word until it is recognized (regardless the reliability level reached).

Grammars=CDEF0123456789
The grammars to activate in the calibration procedure.

AnnounceOk=Valid
The voice announce to user when the word is valid (good response and good reliability level).

AnnounceNER=Recognized
The voice announce to user when the word is a good response but a not enough reliability level.

AnnounceER=Invalid
The voice announce to user when the word is a not a good response.

Welcome=Voice calibrate procedure.
The phrase said when the procedure starts.

Goodbye=End of calibration.
The phrase said at ending the procedure.

7.2 Voice Triggers

Voice triggers are several sections that allow to perform voice actions depending on conditions to match on the emulated screen. Each trigger is composed of 3 sections:

- The conditions section (conditions to be satisfied in order to run the trigger actions): [VOICETRIGGER_XX].
- The TTS section (TTS actions to do): [VOICETRIGGER_TTS_XX]
- The ASR section (ASR actions to do): [VOICETRIGGER_ASR_XX]

One (at least) or more triggers can be defined.

Trigger 1 …
[VOICETRIGGER_01]
[VOICETRIGGER_TTS_01]
[VOICETRIGGER_ASR_01]

Trigger 2 …
[VOICETRIGGER_02]
[VOICETRIGGER_TTS_02]
[VOICETRIGGER_ASR_02]

Trigger 3 …
[VOICETRIGGER_03]
[VOICETRIGGER_TTS_03]
[VOICETRIGGER_ASR_03]
…

7.2.1 Voice trigger conditions section

The conditions to match on the emulated screen to run the trigger should be defined on this section.
There are two kinds of conditions: cursor position, on one hand, and three text conditions, on the other. Text conditions are excluding conditions (Logical AND), i.e. all the text conditions must be true to run the trigger.

[VOICETRIGGER_XX]
On=Yes
Enables/ disables this trigger.

Up to 5 text matches on the emulated screen

Type_1=A=
Type of match on screen: A Alphabetic, N Numeric.

- Equal (A=, N=)
- Not equal (A!=, N!=)
- Greater (A>, A<)
- Lesser (A<, N<)
- Greater or equal (A>=, N>=)
- Lesser or equal (A<=, N<=)

Row_1=4
Row position where to look for a text.
Row=0 and Col=0 means 'unused condition'.
Col_1=4
Column position where to look for a text.
Row=0 and Col=0 means 'unused condition'.

Match_1= Any mix of {R,C,L}, variables {#1} {#2}{#$}, or expressions {{1+(2*#1)}}
Text to look for at Row_1 / Col_1 position. To consider ending spaces, text should be delimited by quotes “”.
If no text is supplied, the condition is not evaluated (TRUE).
To compare two areas in screen set here an {r,c,l} tag to compare the text denoted by {r,c,l} against the text located at Row_1, Col_1 position.

Cursor position condition

CursorAtRow=8
Cursor row position to evaluate. 0 (zero) means in any row.

CursorAtCol=20
Cursor column position to evaluate. 0 (zero) means in any column of the row.
Row=0 and Column=0, means anywhere in the screen.

Previous=!2&!3&!4|=5|=7
=5 Valid if previous is 5
!5 Valid if previous is NOT 5
It is possible to combine by and (&) / or (|)
!2&!3&!4
The previous trigger must be Neither 2 nor 3 nor 4
=5|=7
The previous trigger must be 5 or 7

HideScreen=No
Do not display the PDA's screen while the trigger is running.

HideColor=Green
The background color to display on-screen when the hiding screen is performed.

HideArea=
Hides specific areas of the screen. Set with one or more {r,c,l} tags ({row,column,length}).

Scanner=0-DoNothing
Orders the PDA's scanner to (1) activate or (2) deactivate reading.

LockKbd=0-DoNothing
The PDA's keyboard is (1) Locked (2) Unlocked.

7.2.2 Voice trigger TTS actions section

[VOICETRIGGER_TTS_XX]

These are the Text-To-Speech actions to do if the screen conditions match the voice trigger conditions. Constant text (supplied in the trigger) and variable text (extracted from emulated screen) will be said to user.

On=Yes

Enables / disables TTS actions in this trigger.

TtsSay01=Any mix of {R,C,L}, variables {#1} {#2}{#$}, or expressions {{1+(2*#1)}}

Combination of constant text (normal text in key) and variable text ("RCL", variables or calculations).

RCL text is indicated by three numeric values denoting the starting point of variable text by row and column, and its length "{r,c,l}".

All the text is read out loud, except for those parts surrounded by special spell tags.

Special text tags <SPL><SAY> delimit text to spell, i.e.the text "location <SPL>AB8<SAY> to go" will be read as "location *y be eight to go*".

The last value (l) in the {r,c,l} tag may be preceded by a letter to indicate special operations:

- {r,c,Tl} means alphabetic spaces trim (no spaces)
- {r,c,Nl} means numeric trim (no leading zeroes)

The {r,c,l} tag can have an additional operator to handle some options {r,c,l,o}

- {r,c,l,Lo} means Left suppression of "o" characters. I/e {3,5,N20, L2) will take at row 3 and column 5 twenty characters, then suppress leading zeroes by numeric trim, then suppress leftmost 2 characters.
- {r,c,l,Ro} means Left suppression of "o" characters. I/e {3,5,T20, L2) will take at row 3 and column 5 twenty characters, then suppress spaces by trim, then suppress rightmost 2 characters.

Conditionally say

The starting tag <X> allows to **conditionally say** the line depending on the previous trigger execution:

- <=> The same trigger. I/e: "TtsSay01=<=>Loop on the same trigger" will be said to user only if the previous trigger was the current one.
- <!> Not the same trigger. I/e: "TtsSay01=<=>New trigger" will be said to user only if the previous trigger was another than current.
- <=3> The trigger 3
- <!5> Not the trigger 5.

Data storage Variables

Data **storage variables** allow manipulating data over one trigger to another.
10 data storage slots are available from #0 to #9.
These variables can be set on the TTS Trigger section from constant or dynamic (screen by {r,c,l}) content. One line of TtsSayXX is used to fill the variable i/e:

```
TtsSayXX=<#1> Constant text to say about {10,3,9}
```

The line must start with the variable token <#x>, and the text is not said, but used to fill the variable.
These variables can be used on the TTS Trigger section into the text to say into curly brackets {#0} to {#9} i/e:

```
TtsSayXX=These screen and stored data {#4} from others.
```

Expressions

The four arithmetic operations (+-*/) can be done over on these variables (with constants or other variables) when they are used, the result can be said or used to fill another variable like:
{{#x+#y}} {{#x+Y}} {{Y/#3}}{{#7*3}}…
Other variables or constants can be used in arithmetic operations:

```
TtsSayXX=<#1>{{#3+1}}
TtsSayXX=Quantity to pick is {{#1+#2}}
```

See chapter 7.2.4 about expressions for more details.

Automatic triggers “auto triggers”

An automatic trigger is denoted by a grammar “$” in the ASR section.
In this case the TTS is not said to the user and stored in a variable.

```
TtsSayXX=These text is stored.
```

The TTS text to say can be used after in another trigger by using the {$} syntax.

```
TtsSayXX=Old text is {$}.
```

If you need to say the current trigger at the moment, use this syntax

```
TtsSayXX=<#$>This is the text said on current trigger
```

TtsSay02= Any mix of {R,C,L}, variables {#1} {#2}{#$}, or expressions {{1+(2*#1)}}
Several TTS actions are allowed.

7.2.3 Voice trigger ASR actions section

[VOICETRIGGER_ASR_XX]
These are the ASR actions to perform in this trigger:

On=Yes
Enables / disables ASR for this trigger.

Grammars=CDEF0
Grammars to use in this trigger. A grammar is a set of rules and words that ASR engine use for speech recognition. The grammars are defined in cd_xx.bnf file that can be customized. Standard ASR engine includes three grammars:

- "D" for **digits** (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
- "E" for **digex** (00 - 99).
- "C" for **commands** (Cancel / Accept)
- "F" for **functions** (function keys keywords)
- "0" to "9" for **custom** grammars.

Other grammars (suspend / resume / calculator) are internally managed by ASR engine.

LenMin=2
Minimum voice input length expected. Inputs shorter than this will be rejected, and a new input will be requested.

LenMax=4
Maximum voice input length expected. Inputs bigger than this will be rejected, and a new input will be requested.

Confirm=Yes
Set ***Confirm=***"No" to use the ***Simple*** voice input algorithm to perform ASR input action.
Set ***Confirm=***"Yes" to use the ***Validate*** voice input algorithm to perform ASR input action unconditionally.
Set ***Confirm=***"Check" to use the ***Validate*** or ***Simple*** voice input algorithm to perform ASR input action following check conditions (see Check).
See Input Modes paragraph for an explanation of ASR algorithms.

ConfirmDigits=%s is correct?
Question that will be asked to user "Confirm" mode.
The "%s" tag is replaced by the input value.

Repeat=Yes
Activates Repeat algorithm to perform ASR input action.
See Input Modes paragraph for an explanation of ASR algorithms.

SayOneTime=Yes
The voice announce is said before any ASR recognition. The if ASR recognition is not valid, the voice announce will be automatically repeated (every 10 seconds approximately) or not (set SayOneTime=Yes).

CancelDo=Yes
Allows user to cancel a data input, and sends cancel sequence to host.

Cancel=0123
Sequence sent to host when the user cancels the input without any data.

CancelConfirmation=please confirm %s
Question that will be asked to user if a Cancel command is said in "Confirm" mode.
The "%s" tag is replaced by the input value.

Check=Text or {r,c,l,o}
The user input may be checked against this value to be validated. If not equal, the input is rejected and the ASR is started again. If equal, the input is validated, and sent to host. This is suitable to validate user input that must match any data on screen (I/e 3 last digits of a product code).
The check is defined by a 4 operators tag {r,c,l,o}
r= Row, c=Column, l=Length, o=Match operation.
The operation can be Mn (Match n digits) or Xn (Checksum match on n digits)

- {4,7,N15,M3} will take 15 characters of text at row 4, column 7, perform a numeric trim, then match the last 3 characters on user input.
- {4,7,T15,M3} will take 15 characters of text at row 4, column 7, perform a trim, then match the last 3 characters on user input.
- {4,7,15,X3} will take 15 characters of text at row 4, column 7, perform a check-sum, then match the last 3 characters on user input. The check-sum algorithm is specific to voiXtreme, an utility is available to calculate this algorithm. This method will be suitable to generate pseudo-random verification digits (i/e on locations codes).

CheckDo=R

- "R" to reject the input if not equal (backward compatibility)
- "=" must be equal
- "!=" or "<>" Must be different
- ">" Must be greater
- "<" must be lower
- "<="Must be greater or equal
- ">=" Must be lower or equal

Format=Any mix of {R,C,L}, variables {#1} {#2}{#$}, or expressions {{1+(2*#1)}}

The output text can be formatted to add fixed or variable text to user input.

- The {$} tag represents the user input.
- Some {r,c,l} tags can add text form the display
- Constant text can be added.

On user input, initializes two variables:

- #$ which contains the user's raw input;
- #& which contains the user's formatted input

Example: Format=9999999{$},00{\09}{4,3,5} will prefix user input by constant text "9999999", then follow it by constant text ",00" then a tabulation char, and finish by 5 characters text located at row 4 column 3. I/e if user input is "123" the data sent to host will be " 9999999123,00<TAB>ABCDE".

CheckBc=

Allows to check the scanner input against constant or screen content.

CheckBcDo=R

R to reject the input, A to accept the input without formatting.

FormatBc=

Output formatting string for barcode read. {$} means the user input.
May contain fixed and variable data. E.g.: 001{$}999{2,15,3}
If empty, no format is done.

Terminator=6

Terminator string that will be sent to host after the ASR formatted data (added as trailing) to validate input (in this case, Field Exit).
A->F1 … X->F24
0->Enter
1->Clear
2->Page Down
3->PageUp
4->Forward Tab

5->Back Tab
6->Field exit

AltTermCode=6
Terminator code may be changed by user keyword locution (AltTermKw) during input.
Alternate Terminator AID CODE that will be sent to the host with the ASR data.
See "**Terminator**" for values.

AltTermKw=
Keyword used to switch to alternate terminator mode.
Set a keyword in this configuration key to activate the feature.
If this keyword is recognized, the Terminator code will be changed by **AltTermCode.**
Check that the keyword is present in an active grammar for this trigger.

AltTermPhr=Alternative mode
Phrase that will be said when switching to alternative terminator mode.

Expected=
Expected data for this input, allowing to validate with lower reliability levels.
Combination of constant text (normal text in any key), and variable text (delimited by curly brackets {row,column,lenght}).

Information=
The sentence to say to user if the **ResInfo** keyword is recognized. This sentence is intended to help the user about actions to do on this trigger in case of help (i/e "go to specified location then say the check digits", or "take the announced quantity then validate it.

CponfirmInfo=
The sentence to say to user if the **ResInfo** keyword is recognized, during the confirmation algorithm. This sentence is intended to help the user about validation of data (i/e "say accept to validate or cancel to say another quantity").

FormatKw=
Output formatting string for Kbd keyword FFFF.
May contain fixed and variable data. E.g.: 999{2,15,3}.
If empty, no format is done.

ListenOnTts=No
Enables the user to interrupt the TTS locution by speaking.

7.2.4 Expressions

Since TelNet version 4.4.0, it is possible to add calculations in text strings used for trigger conditions, TTS and ASR.

These calculations are called expressions and their syntax obey the following rules:

- No white space, not even at the start or at the end;
- All values must be unsigned integrals;
- Operator precedence must be enforced by enclosing operations inside of a pair of parenthesis;
- All operators are binary operators. To get the behavior of the unary minus, use “0-value” where “value” is the value to get the opposite of.
- The syntax of variables is “#N” where “N” is either the number of the variable (“0”, “1”, etc), or a symbol representing a special variable (“$” for last voice raw input, “&” for last voice formatted input);
- If the number of a normal variable (#1, #2, etc) is not a constant (i.e the result of another expression), it must be enclosed within parenthesis : “100+#(#1*2)”;
- Expressions cannot be used with {R,C,L} values.

Valid examples	Invalid examples
1+(3*(1000/#1))	1+2+3
1+(1+(1+(1+1)))	# 12
(#95+#$)^3	1+(1+2))
0-#1	1+((1+2)
((((15))))	()
10+#(#1+#&)	1 + 2
(1+15)	1+2

Available operators:
+
-
* (multiply)
/ (divide)
% (modulo)
^ (power)
WARNING: When using expressions, the top-level (depth 0) expression must be enclosed within a second level of brackets.

Valid examples:

- `Match_1=Some text {1} some text {{#1+10}} some text`
- `Match_1=Some text {{1}} some text {{#1+10}} some text`
- `Match_1=Some text {#1} some text {{#1}} some text`
- `Match_1=Some text {1,2,10} some text`

Invalid examples:

- `Match_1=Some text {1} some text {#1+10} some text`
- `Match_1=Some text {1} some text {{#1+{(10+#2)}}} some text`
- `Match_1=Some text {1} some text {{#1+{10+#2}}} some text`
- `Match_1=Some text {1,2,{10+#1}} some text`

It is recommended to always use double brackets, even when it is not necessary, to keep a uniform syntax.

Such expressions can be used in:

- "Match_N" in [VOICETRIGGER_N]
- "TtsSay_N" in [VOICETRIGGER_TTS_N]
- "Check" in [VOICETRIGGER_ASR_N]
- "Format" in [VOICETRIGGER_ASR_N]
- "CheckBc" in [VOICETRIGGER_ASR_N]
- "FormatBc" in [VOICETRIGGER_ASR_N]
- "Expected" in [VOICETRIGGER_ASR_N]
- "Information" in [VOICETRIGGER_ASR_N]
- "ConfirmInfo" in [VOICETRIGGER_ASR_N]
- "FormatKw" in [VOICETRIGGER_ASR_N]

WARNING: "{$}" and "{#$}" represent two different things. $ is the very last input. For example, if the use has to pronounce a number, and then say "confirm" to validate it, then $ will not be the number he said, it will be the code associated to the keyword "confirm" instead. #$ however represents the last "real" user input, i.e the number in the previous example. {$} should only be used for user input formatting. If one needs to get the value of the previous user input, the variable #$ should be used instead.
Example:

- Match_1={#$}
- Match_1={{#$+10}}
- TtsSay_1=Last user input plus 10 was {{#$+10}}
- Format={#$} becomes {$}

WARNING: The variable #$ is initialized before any formatting on the user input is done. In the last example, if the user said, for example, 12 as first input, and 10 as second input, then the formatted second input will be "12 becomes 12", not "10 becomes 12"

7.2.5 Extended expressions

As of version 4.4.9, “extended” expressions are allowed in the property “DoIf” of a trigger’s match section.

Expressions are the same as the ones used until now, but with extra features :

- Handling of 3 value types : integral, decimal, and text;
- Conversions between numeric and text types;
- New operators.

Like previous expressions, the extended expressions have to obey to certain rules:

- Operator precedence is not handled and a pair parenthesis is mandatory **every time we want to use more than a single operator on the same depth level**;
- White space characters are now allowed;
- Variables identifiers are prefixed with “#” and can contain any characters in [A..Z], [a..z] and [0..9];
- Special variables representing the previous VoiXtreme inputs (“#$” and “#&”) are written “#LRI” and “#LFI” for “Last raw input” and “last formatted input” respectively. This new syntax is also handled by the old expressions as of this version.

Valid examples	Invalid examples
(0 < #LRI) & (#LRI < 100)	0 < #LRI & #LRI < 100
123 << (‘456’ << 789)	123 << ‘456’ << 789
10 + ‘10’	10 + ‘non-numeric’
(((‘abc’ << 3.14)))	()
#1	# 1
! #A	0 & #A

Available operators are :

- "**<<**" Concatenation. Always outputs a string value. Input values can be of any type and will be converted to string. The right operand is appended after the left one ;
- "**+**", "**-**", "*****", "**^**" Addition, substraction, multiplication and exponent. Input values can be of any type, even string, as long as they can evaluate to numeric values (i.e " '123' " is allowed but " '123$' " will cause an error). The result is always a numeric value. If at least one of the two operator is or evaluates to a decimal value, the result will also be a decimal value. If both operands are or evalute to integral values, the result will also be an integral value.
- "**/**" Division. Same as multiply operator "*", but the right operand must not be or evaluate to a null value (i.e "0" or "0.0").
- "**%**"Modulo. Same as division operator "/", but both operand have to be or evaluate to values of integral types. The right operand must not be or evaluate to a null value (i.e "0" or "0.0"). The type of the output value will always be integral;
- "**&**" and "**|**" Boolean AND and OR. Operands are converted to their boolean equivalent. Any non-empty string or value that is or evaluates to a non-null numeric value is considered true, and all other values evaluate to false. The result will always be of integral type with either 1 or 0 for true and false respectively. The second operand is evaluated only if the first one is not enough to determine the result of the boolean operation;
- "**<**", "**<=**", "**>**", "**>=**", "**=**", "**!=**" Smaller, smaller or equal, bigger, bigger or equal, equal, different. Can be used to compare values. If both of the operands are of type string, they will be compaired alphabetically, else the string operand will be converted to the equivalent numeric value. If one of the two operands is or evaluates to a decimal value, the decimal type will be favored to avoid information loss due to number rounding (which means that "0.9 < 1" is true). The operators "=" and "!=" do not compare the values types (which means that " 1 = 1.0 ", " 1 = '1' " are both true).

7.3 Sample configuration file

8 Development of voice applications for WireLess TelNet 5250

8.1 Description

The system of "Triggers" is adapted to perform operations with voice and screen transactions already developed in text mode.

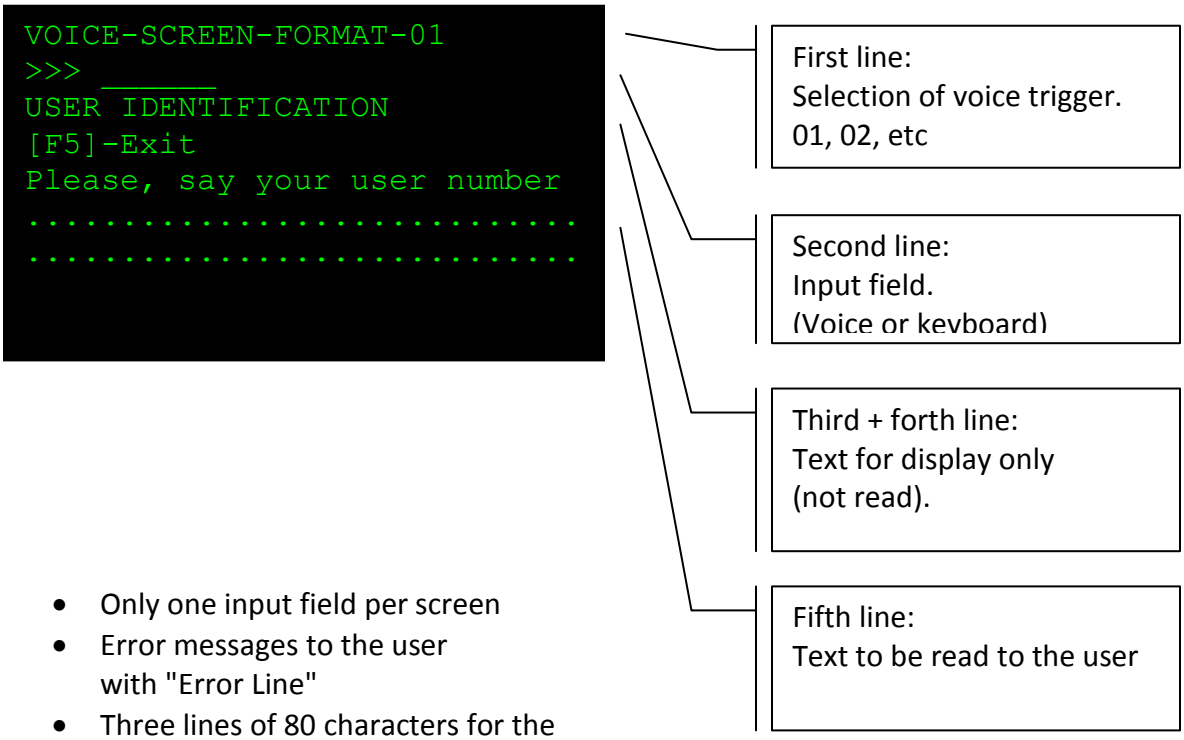
For each input field it is necessary to develop a "Trigger" voice so as to trigger the voice system at the right time with the correct parameters.

This has the disadvantage of having an application developed in two parts, the screens on the AS400 (programming) and triggers on the PDA (configuration file).

If we wish to develop a voice application with a specific interface 5250 we can configure a small number of generic triggers (one or two), and develop the screenshot the 5250.

8.2 Pattern

Our 5250 screen may have the following pattern:



- Only one input field per screen
- Error messages to the user with "Error Line"
- Three lines of 80 characters for the text to read.
- Management of the function keys.
- Display for development assistance and testing.

8.3 Corresponding configuration file

```
// 5250
//-------------------------------------------------
// Tts definitions
[TTS_DEF]
VoiceSpeed=70
VoiceVolume=70
TtsPrinterOn=No
TtsErrorOn=Yes
MsgWelcome=Welcome to Telnet by soff to go.
MsgDisconnect=End of connection.
MsgConnected=Connected.
MsgConnecting=Connecting.
KeyRepeat=0000
KeyVolUp=0000
KeyVolDn=0000
KeySpeedUp=0000
KeySpeedDn=0000

//-------------------------------------------------
// Asr definitions
[ASR_DEF_TUNNING]
AsrOn=Yes
ReliabilityLvl=2700
ReliabilityPause=5500
ReliabilityCmd=4500
ReliabilityExp=300
AsrThreshold=50
MinSpeechDur=240
TrailingSilence=100
TimeoutSpeech=5000
SyncTts=Yes
StartBeep=\windows\voixtreme.wav
StartBeepPause=\windows\voixtreme.wav
LogDo=Yes
LogFile=\WTnAsrLog.log

[ASR_DEF_GRAMMAR]
CustomGrammar0=<custom0>
CustomGrammar1=<custom1>
CustomGrammar2=<custom2>
CustomGrammar3=<custom3>
CustomGrammar4=<custom4>
CustomGrammar5=<custom5>
CustomGrammar6=<custom6>
CustomGrammar7=<custom7>
CustomGrammar8=<custom8>
CustomGrammar9=<custom9>

[ASR_DEF_KEYWORDS]
ResRepeat=repeat
ResAccept=ok
ResCancel=cancel
```

ResPause=speech-pause
ResResume=speech-resume
ResInfo=information
SpeedUp=speed-up
SpeedDn=speed-down
VolumeUp=volume-up
VolumeDn=volume-down
AsrSuffix=ready
KeywordFn01=January|first
KeywordFn02=February|second
KeywordFn03=March|third
KeywordFn04=April
KeywordFn05=May
KeywordFn06=June
KeywordFn07=July
KeywordFn08=August
KeywordFn09=September
KeywordFn10=October
KeywordFn11=November
KeywordFn12=Dercember
PauseInfo=In pause. Say speech-resume to coninue.

[ASR_DEF_KBD_KEYWORDS]
Count=3
Scc00=0018
Kwd00=disconnect
Scc01=0001
Kwd01=primo
Scc02=0002
Kwd02=secundo

[ASR_DEF_CALIBRATE]
TargetReliability=5000
TargetOk=5
SequenceMode=0
SkipOnEr=Yes
Grammars=DCF0123456789
AnnounceOk=Valid
AnnounceNER=Recognized
AnnounceER=Invalid
Welcome=Voice calibrate procedure.
Goodbye=End of calibration.

```
//-----------------------------------------------
// Triggers
//-----------------------------------------------

//-----------------------------------------------
// Voice trigger "Format 01"
[VOICETRIGGER_01]
On=Yes
Row_1=1
Col_1=1
Match_1=VOICE-SCEEN-FORMAT-01
Row_2=0
Col_2=0
Match_2=
Row_3=
Col_3=
Match_3=
CursorAtRow=2
CursorAtCol=5

// Voice trigger Tts
[VOICETRIGGER_TTS_01]
On=Yes
TtsSay01=input {5,1,80}.
TtsSay02= input {6,1,80}.
TtsSay03= input {7,1,80}.

// Voice trigger ASR
[VOICETRIGGER_ASR_01]
On=Yes
Grammar=1
LenMin=2
LenMax=4
Confirm=Yes
Repeat=Yes
CancelDo=Yes
CancelConfirmation=Do you want to cancel?
Cancel=0123
Terminator=6
```

Note sur les grammaires BNF

Lors de la définition d'une prononciation PRONAS ou phonétique (L&H) pour un « terminal » (mot), cette définition supprime les variantes contenues dans le dictionnaire.

Il est donc préférable de conserver les variantes du dictionnaire, et de créer un « alias » avec le pronas.
Le préfixe xlt_XXXX permet de créer des alias sur les mots.
Du coup je peux avoir un mot avec une vocalisation standard, PLUS une vocalisation spécifique.

Par exemple

!pronounce xlt_pause PRONAS « pauze »

<custom1> pause | xlt_pause ;

Ceci permet de NE PAS substituer les variantes de « pause » et d'ajouter une prononciation spécifique (pauze).
La librairie VoiXtreme fera le traitement pour assimiler le « xlt_pause » à « pause ».