



## SPELL GUI Manual

*Software version 2.0*

<i>Author</i>	<i>Date</i>	<i>Version</i>	<i>Comment</i>
Rafael Chinchilla & Fabien Bouleau	13 April 2010	3.0	Document created
J. Andrés Pizarro	20 September 2010	3.1	Added call stack, breakpoints and text search feature description. Workbench layout description revised.
J. Andrés Pizarro	12 October 2010	3.2	Added Outline and Variables views description.
J. Andrés Pizarro	22 November 2010	3.3	Update software target version. Configuration section reviewed. Added shell view section. Added help section.
Rafael Chinchilla	22 Nov. 10	3.4	Added procedure properties Added prompt blinking feat. Added procedure properties dialog. Minor amendments

Prepared By	Rafael Chinchilla Fabien Bouleau J. Andrés Pizarro	Signature: <i>signature on file</i>
Reviewed By	Thomas Nowak	Signature: <i>signature on file</i>
Authorized By	Thomas Nowak	Signature: <i>signature on file</i>

# Table of Contents


<b>1</b>	<b>INTRODUCTION .....</b>	<b>6</b>
1.1	Purpose of this document .....	6
<b>2</b>	<b>FRAMEWORK ARCHITECTURE.....</b>	<b>7</b>
<b>3</b>	<b>SPELL GUI COMPONENTS .....</b>	<b>8</b>
3.1	The navigation view.....	8
3.1.1	Getting procedure properties.....	9
3.2	The procedure view .....	10
3.2.1	The Tabular presentation .....	11
3.2.2	The Text presentation .....	13
3.2.3	The Shell presentation .....	14
3.2.4	Presentations area .....	14
3.2.5	The control area .....	15
3.2.6	Prompt inputs .....	15
3.2.7	Other features.....	16
3.3	The master view .....	17
3.3.1	Master console .....	18
3.3.2	Log viewer .....	18
3.4	The call stack view .....	18
3.5	The Outline view .....	20
3.5.1	Interacting with the outline view .....	21
3.6	The Variables view.....	21
3.6.1	Checking variable values .....	21
3.6.2	Watch of variables.....	22
3.6.3	Value modification .....	23
3.7	The status bar .....	23
3.8	The connection dialog .....	24
3.9	The executors dialog.....	25
<b>4</b>	<b>STARTUP AND CONFIGURATION .....</b>	<b>26</b>
4.1	Startup .....	26
4.2	Configuration .....	26
4.2.1	Configuration file.....	26
4.2.2	Preferences management .....	29
4.2.3	Exporting and importing preferences .....	30
4.3	Initial steps .....	30

<b>5</b>	<b>PROCEDURE EXECUTION .....</b>	<b>31</b>
5.1	Procedure execution status .....	31
5.2	Foreground and background procedures.....	31
5.3	Procedure execution control.....	32
5.4	Procedure input .....	32
5.5	Manual goto.....	33
5.6	Monitoring and control .....	33
5.6.1	User handover.....	34
5.7	Scheduling procedures.....	34
5.8	Procedure files.....	36
5.8.1	As-Run files .....	36
5.8.2	Log files .....	36
5.9	BREAKPOINTS.....	37
5.9.1	Adding or removing permanent breakpoints .....	37
5.9.2	Executing a procedure until a line is reached (temporary breakpoints).....	37
<b>6</b>	<b>OTHER FEATURES .....</b>	<b>38</b>
6.1	Printing .....	38
6.2	Code search .....	39
6.3	Copy source code.....	39
<b>7</b>	<b>GETTING HELP.....</b>	<b>40</b>

## Ref. Documents

## Acronyms

CV	Command Verification
GCS	Ground Control System
GDB	Ground Database
GUI	Graphical User Interface
HMI	Human Machine Interface (equivalent to GUI)
IDE	Integrated Development Environment
MMD	Manoeuvre Message Database
OOL	Out-of-limits
PDF	Portable Document Format
PROC	Automated SPELL procedure
RCP	Rich Client Platform
S/C	Spacecraft
SCDB	Spacecraft Database
SDE	SPELL Development Environment
SEE	SPELL Execution Environment
SES	Société Européenne des Satellites
SPELL	Satellite Procedure Execution Language and Library
TC	Telecommand
TM	Telemetry
URI	Uniform Resource Identifier
USL	Unified Scripting Language
UTC	Coordinated Universal Time

13 April 2010	SPELL GUI Manual	
Page 6 of 40	SPELL	
	Software version 2.0	
	UGCS-USL-SPELL-GUI-SUM_08_003_3.4.doc	

# 1 Introduction

## 1.1 Purpose of this document

This document is the software user manual for the SPELL GUI client application. The SPELL GUI allows SPELL users to connect to a SPELL server in order to load and control the execution of satellite procedures. In a nutshell, the operations that can be performed on the SPELL GUI are:

- Connect to a SPELL server
- Connect to and manage SPELL contexts
- Load, close, schedule or kill SPELL procedures
- Control and monitor procedure execution

The SPELL GUI is a Java/RCP application, available in both GNU/Linux and Windows® platforms.

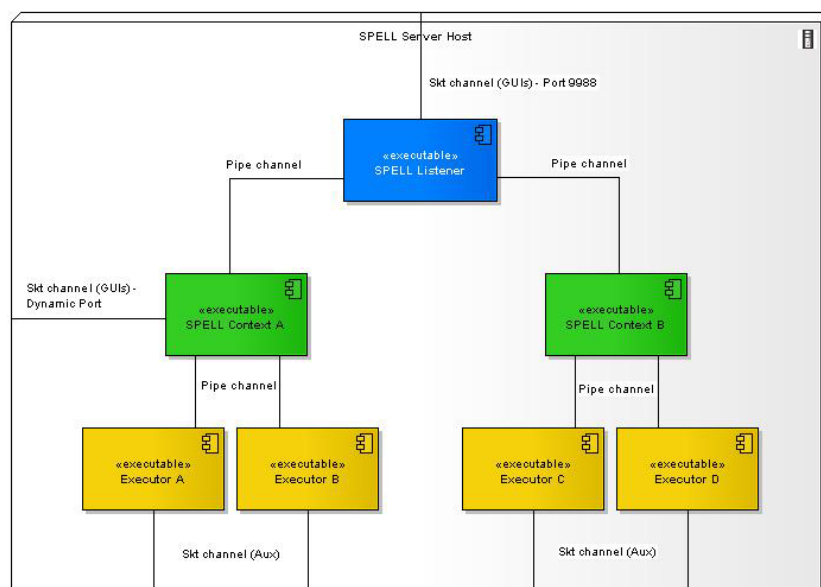
## 2 Framework architecture

The SPELL architecture can be divided into two parts, the *SPELL execution environment (SEE)* and the *SPELL development environment (SDE)*. They can be also seen as the on-line and the off-line part of the SPELL framework. In this document we focus on the SEE components.

The execution environment includes all the elements needed for executing procedures. The main components of this environment are:

- (a) the *SPELL server*, core of the environment, where the procedure execution is performed. It is responsible of coordinating all tasks, interfacing with ground control systems, etc.
- (b) the *SPELL GUI clients*, graphical interfaces through which the procedure executions are controlled and supervised by S/C controllers and engineers.

The SEE architecture is based on a TCP client-server philosophy. The overall structure of the SPELL server can be seen below:



**Figure 1: Structure of the SPELL server**

The SPELL GUI clients first connect to the *SPELL listener* process. This is the entry point of the server, and it coordinates and registers all connected clients.

Once a GUI is connected, the listener provides to the GUI with a list of all available *SPELL contexts*. A context is a separate process (colored in green in the picture) which is in charge of controlling and coordinating the execution of SPELL procedures for a particular spacecraft. Each context encapsulates data, communication and control for only one spacecraft: procedures running within a given context can interact and share data, but they cannot communicate with procedures running on a different context. Data files are also hidden from procedures not running in the same context.

The list of available contexts include the context status, that is, for each context it indicates whether it is running or not. If the desired context is not running, it can be started from the SPELL GUI interface. Once the context is ready, the SPELL GUI needs to be *attached* to the context. By doing so, a direct connection is established between the GUI and the context. At this point, the GUI can start working with SPELL procedures.

## 3 SPELL GUI components

The SPELL GUI is an Eclipse RCP-based application. Therefore, its design follows the Eclipse IDE guidelines and concepts, and its appearance is alike to an Eclipse IDE. The main GUI components are Views. A view is an inner and independent window which shows specific information and allows the user to work with specific parts of the application. In addition to the views, the GUI provides a menu and toolbar for triggering actions, and some dialogs for feature configuration or for performing certain tasks.

The GUI is composed of the following views:

- Navigation view
- Master view
- Call stack view
- Outline view
- Variables view
- Procedure views

A general view of the SPELL GUI can be seen below. Note that in this snapshot, no procedures have been open yet; therefore no procedure views can be seen. Also notice both Navigation view and call stack view are both stacked in the navigation area.

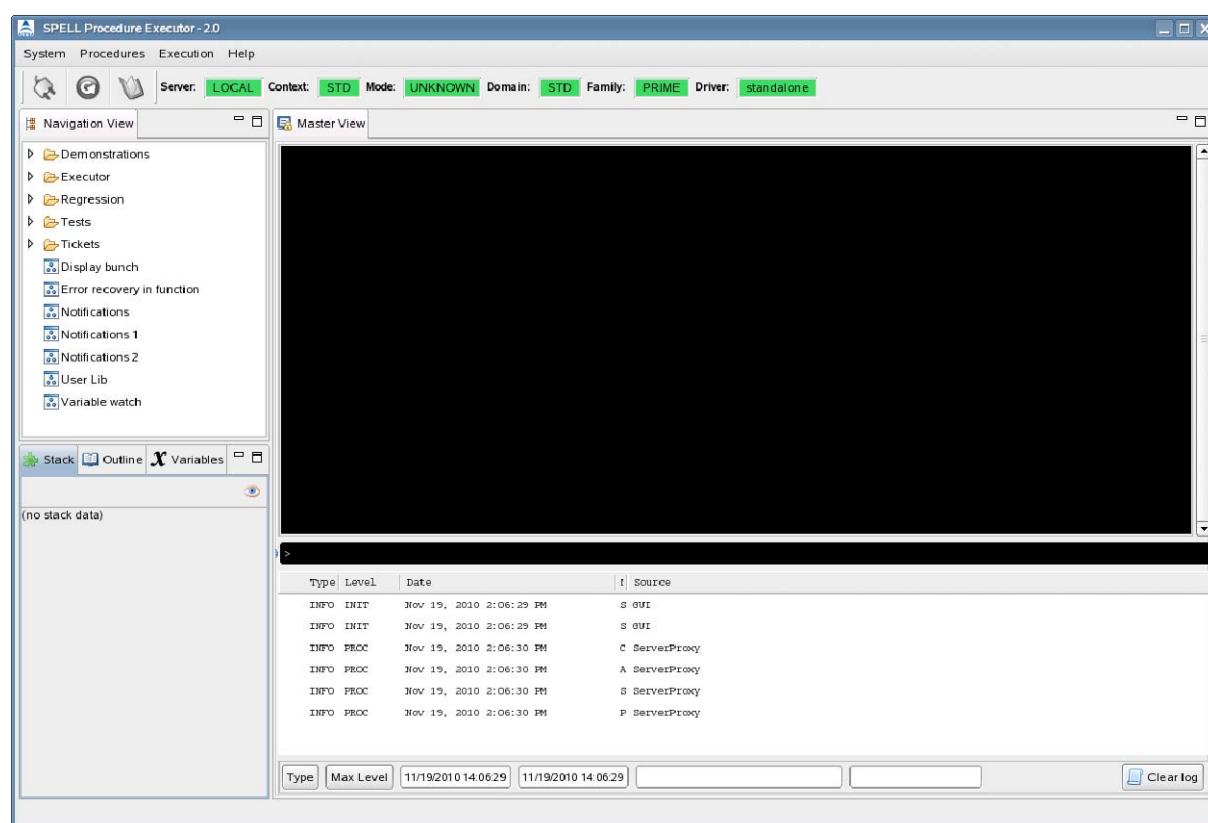


Figure 2: General view of the SPELL GUI

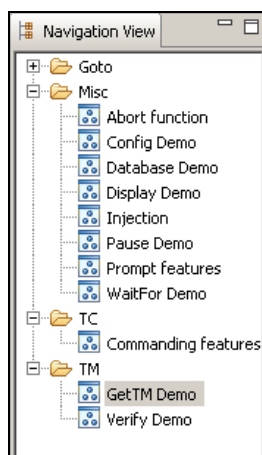
### 3.1 The navigation view

The navigation view shows the set of available procedures once the application is connected to a SPELL Context.



The list of procedures consists on a list of procedure names organised in folders. The folder structure reflects the directory structure of the procedure base directory located at the SPELL server host. This directory is specified in the SPELL context configuration file (its location is context-specific).

An example of navigation view can be seen below.

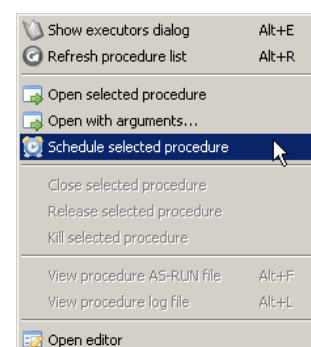


**Figure 3: Navigation view**

The navigation view is the fastest way to start a procedure: by double-clicking on the procedure name in the view, the procedure will be loaded with default settings.

Also, it is possible to select a procedure from this list, and then go to the menu *Procedures* and select one of the available options:

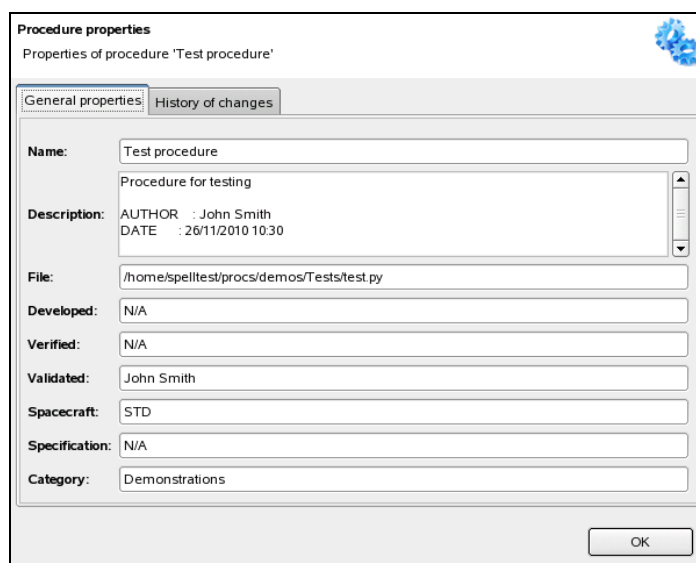
- *Open selected procedure*: will open the selected procedure in the navigation view with default settings.
- *Open with arguments*: will allow the user to specify procedure argument values before opening the procedure.
- *Schedule selected procedure*: will open the selected procedure in schedule mode. The user may specify a time or telemetry condition for the schedule.



The procedure list in the navigation view is not automatically updated if ever new procedure files are created in the server side, or existing ones are deleted. To catch up these changes, the "*Refresh procedure list*" button or menu command (in the *Procedures* menu) shall be used. This command will make the GUI to request the context a fresh list of available procedures and display it.

### 3.1.1 Procedure properties

Procedure source files can define some properties in its header, such as the author, versioning information or change history. This information can be seen by right-clicking on a procedure title on the Navigation View, and choosing the menu option "Procedure properties". A new dialog with the procedure information will appear.



**Figure 4. Procedure properties dialog**

In this dialog two tabs are shown. The General properties tab shows general information about the procedure, such as the author, description or target spacecraft this procedure was created, while the History of changes tab helps tracking the changes made along the procedure's lifetime.

## 3.2 The procedure view

The procedure view is the most important view of the application, since it allows the user to watch what is going on during the execution of a procedure. It also provides means for controlling the execution.

There may be none or many procedure views open. Each view is linked to a single procedure instance. Notice that it is possible to open the same procedure several times within the same SPELL Context; each procedure would be called "instance". Each instance is identified by the procedure identifier plus an instance number, starting at zero.

All procedure views are arranged on the GUI workspace area (initially occupied by the Master view only), and they are selectable by clicking on their corresponding title tab.

Procedure views contain one or more different *procedure presentations*. A presentation is a way to show the procedure status, data and notifications, in a particular way. There are two basic presentations available in SPELL, called *Tabular* and *Text*, but more presentations could be added to the application if desired.

The *Tabular*, or code presentation, shows the SPELL procedure source code and a set of three columns where item names, values and status are shown. For example, when a telemetry point value is acquired, the point mnemonic and value are shown in these columns.

The *Text* presentation shows a plain, console-like text output of the procedure. All messages (information, warnings and errors) coming during the procedure execution are shown.

An example of procedure view, with the *Tabular* presentation selected, is shown below:

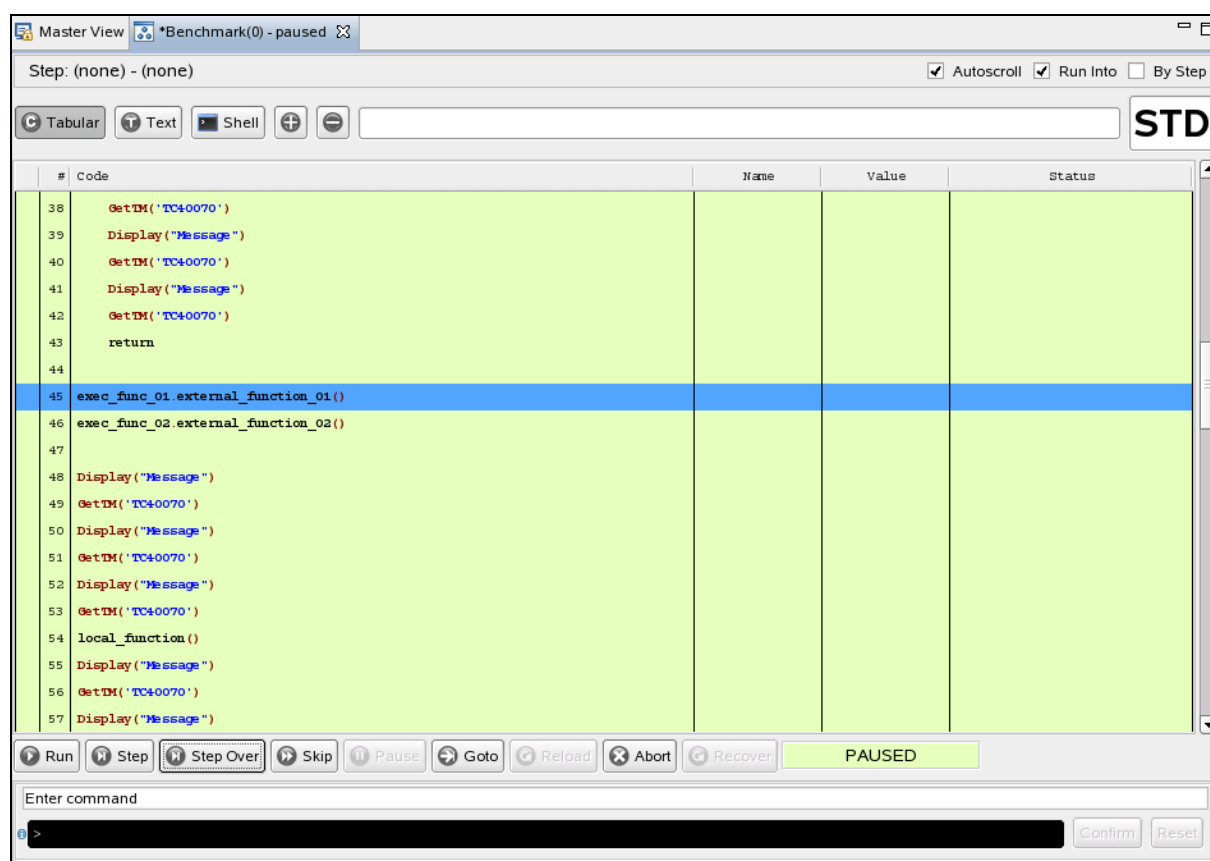


Figure 5: Procedure view example

The main areas in this view are:

- **Presentations area:** containing the presentation selection buttons, the zoom controls, the main message display and the spacecraft indicator. The presentation selection buttons shown depend on the presentations available. In the example above, the Tabular, Text and Shell presentations are installed.
- **Procedure area:** shows the procedure code and/or execution information (the actual contents depend on the presentation selected in the page control area).
- **Execution control area:** provides buttons for controlling the procedure execution, shows the procedure status, and contains the user input area which may be used for gathering user input or for issuing commands to the procedure using the keyboard.

Notice that the view title contains the procedure name, the instance number between parentheses, and the procedure status.

### 3.2.1 The Tabular presentation

As it has been explained, the Tabular or code presentation shows the SPELL procedure source code and a set of three data columns.

Source code lines are numbered, and they are marked as well as they are executed to highlight the parts of the procedure already executed. The current line is marked in bright blue color.

The SPELL language is syntax highlighted in order to increase readability. Besides the source code, three information columns may be seen. These columns show information about the execution of each SPELL statement (if there is any). This information normally consists on:

- Item name (TM parameter name, telecommand name, etc)
- Item value (TM parameter value, telecommand execution phase, etc)
- Item status (Indicates if the statement is success, in progress, timed-out, failed, etc)

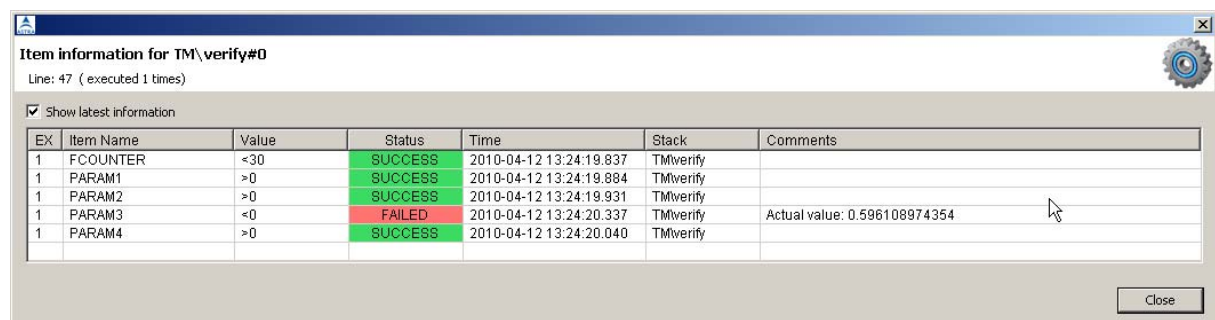
The concrete information shown in these columns depend on the statement being executed. If the SPELL statement is complex and it contains several items (i.e. a multiple TM verification), the information shown in these columns will consist on:

- Nothing on item name column
- Nothing on item value column
- Overall status code and item count for the compound statement: e.g. "IN PROGRESS (5/7)" would mean that the statement information is composed of 7 items, and there are 5 items already processed. Since there are 2 items still in progress, the overall status of the statement is in progress.

Notice the figure on the right. There are three procedure lines that produced item information. The first statement was checking that the TM point 'FCOUNTER' had a value smaller than 30. The check was success. The second statement was composed of the processing of three items, being all correctly processed. The third statement included five items, but only 3 were success.

FCOUNTER	<30	SUCCESS
		SUCCESS (3/3)
		FAILED (3/5)

When information is nested, more information regarding the statement execution is required. For that matter, the *Item Information Dialog* is provided. This dialog is brought up by double-clicking on any of the item information columns, in the desired procedure line:



**Figure 6: Item information dialog**

The dialog provides details about all notifications corresponding to the selected procedure line. The information shown is divided in the following columns:

- **"EX" column:** indicates the number of line execution. Each procedure line can be executed more than one time, and the numbers in this column identify the iteration number. The amount of iterations for a given line appears at the top of the dialog. In the example, "executed 1 times" can be seen.
- **Item name:** the name of the item, which can be a TM point, a telecommand, etc.
- **Item value:** shows the current value of the item, which can be a TM point value, or a telecommand execution stage, among other things.
- **Item status:** the status associated with the value, which can be SUCCESS, IN PROGRESS, FAILED, or other values for command execution stages.
- **Time:** the time of the latest change
- **Stack:** identifies uniquely the procedure line.
- **Comments:** when applicable, they show extra information about the notification.

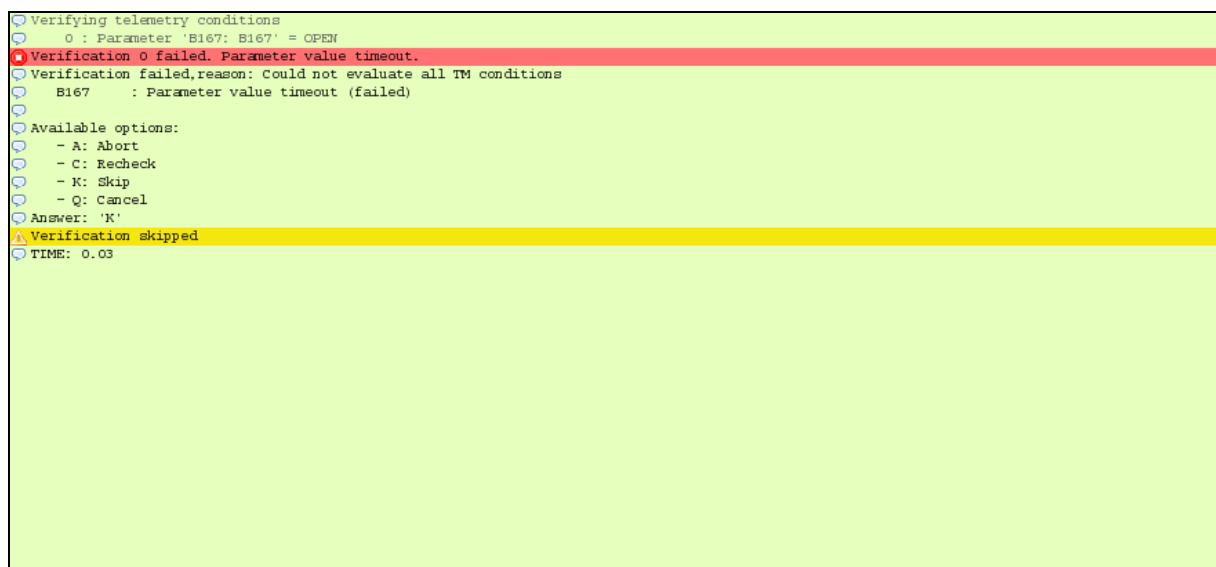
As it has been explained, the item information dialog may show notifications corresponding to *several* line executions or iterations. By default, it only shows the information regarding the latest iteration, but this can be changed to show all notifications by unchecking the check-box "Show latest information" above the notifications table.

The background color of the presentation changes depending on the Procedure status.

### 3.2.2 The Text presentation

The text presentation does not show the procedure source code, but only progress messages regarding the procedure execution. All messages displayed by the procedure are shown in a text area. Messages are identified by their background colour when they are prompt, warning or error messages.

Foreground colors indicate the scope of the messages (according to the preferences selected by the user, see Scope styles on section 4.2.2). Possible scopes are the SPELL procedure itself, the SPELL core and drivers, step instructions, etc. In addition, some messages have an associated icon that to easily identify them.



**Figure 7: the Text presentation**

As in the Tabular presentation, the background color of the text area changes depending on the procedure status.

### 3.2.3 The Shell presentation

The shell presentation allows the user to execute SPELL statements within the Procedure scope. It looks like an operating system console where the user can input SPELL statements on the prompt line, represented by the symbol ">>>". When a valid sentence is entered, the executor processes the code and the resulting output is shown in the next lines.

```
>>> value = GetTM("PARAM1")
Retrieving engineering value of 'PARAM1: Constant parameter'
Last recorded value of 'PARAM1: Constant parameter': 20
>>> Display(str(value))
20
>>> Verify("PARAM4", eq, 20)
Verifying telemetry conditions
0 : Parameter 'PARAM4: String calibration' = 20
ERROR: Verification 0 failed. Parameter value timeout.
Verification failed,reason: Could not evaluate all TM conditions
PARAM4 : Parameter value timeout (failed)

Available options:
- A: Abort
- C: Recheck
- K: Skip
- Q: Cancel

Answer: 'K'
WARNING: Verification skipped
>>> Send("TEST_COMMAND")
WARNING: Failed to execute script: Driver error: Expected keyword: command, group or sequence: unknown (error code 9)
>>> Send(command="TEST_COMMAND")
Sending command 'TEST_COMMAND'
Execution success
>>> |
```

Figure 8: Shell presentation

In the shell presentation any piece of SPELL code, including functions calls, can be written except goto statements which are not valid for manual execution. Multi-line statements like if clauses, for loops, function definitions and so forth are accepted as well. If the shell detects that a multi-line statement is being written, it will help the user doing automatic indentation and the prompt symbol will change to "...". To finish a multi-line statement, a blank line shall be written.

The prompt has command history: by clicking CTRL+Up and CTRL+Down keys the user can recover previously written commands.

It is possible to reset the shell to its initial state and cleanup all output by double-clicking on the shell window.

Notice that any manual execution done from the shell can use and modify the procedure data existing at the current scope. For example, if the procedure execution is stopped within a function call, the code written in the script dialog can access the function local variables. Python scope rules apply as if the manual statements were placed inside that procedure function.

### 3.2.4 Presentations area

The presentations area is always visible, no matter which presentations is chosen, and allows the user to switch between the different procedure presentations. It shows also some presentation control features and gives information about the step a procedure is executing at a time.



Figure 9: Presentations area

The step label shows the current step title and description, if there is any. Next to it there are three check boxes which allow the user to configure the behaviour of the presentations and the procedure execution flow:

- **Autoscroll:** enables or disables the autoscroll feature in all procedure presentations. The actual effect of the autoscroll depends on the presentation: in the Tabular one, disabling autoscroll will make the presentation to show always the same portion of SPELL procedure code, and will not move to show new data appearing somewhere else in the code. In the Text presentation, the text area will not automatically scroll down to show the newest messages.
- **Run Into:** configures the GUI to follow the execution flow by going into all functions and subprocedures. If run-into feature is disabled, the system will *step over* function calls, so that the code inside these functions will not be seen by the user. The run-into feature has no visual effect on the Text presentation.
- **By step:** will make the procedure execution to automatically pause everytime a **Step** is found in the code.



These two last options can be also changed from the Execution menu.

See section 5 for details about the procedure execution status and procedure commands.

At the bottom there are the zoom buttons with the symbols + and - for changing the font size, and a label for showing the latest displayed message. Finally there is a big label showing the attached context.

### 3.2.5 The control area

The control area is always visible, no matter which presentation is chosen, and it allows the user to control the procedure execution.

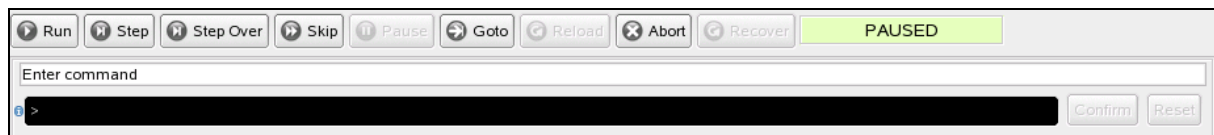


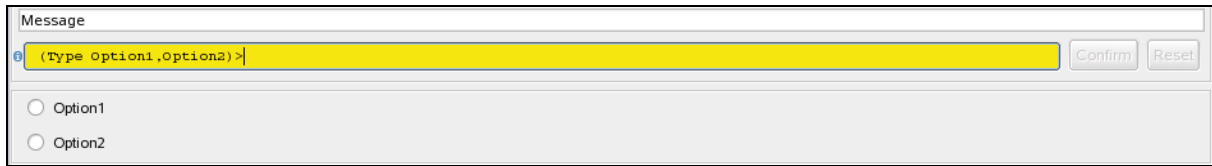
Figure 10: Control area

It provides a set of buttons corresponding to procedure commands like run, step, abort and so on. It also shows the current procedure execution status in a colorized text field.

At the bottom it is shown a text input widget for entering procedure control commands. These commands perform the same actions as the control buttons shown above. There is an information icon at the widget's left which helps entering commands.

### 3.2.6 Prompt inputs

Below the control buttons, the prompt handling area can be found. When there is no active prompt (i.e. the procedure is requesting input) this area shows a black text field where procedure commands can be typed (e.g. "run", "pause", etc.). When there is a prompt active, this area changes in order to show the prompt request.



**Figure 11: Prompt**

Notice that the prompt message is shown above the text field, and that the text field background becomes yellow. An input hint (when available) is also shown in the text field.

When the prompt is of LIST type, a set of radio buttons are displayed below the text field. The user can provide the answer via the text field or by selecting one of the radio buttons.

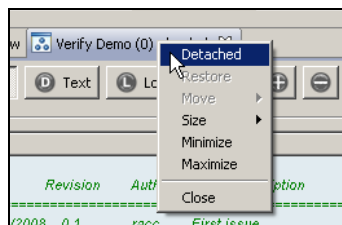
Once the answer is selected, the "Confirm" button has to be clicked to commit the response to the procedure and the execution will continue.

Two extra buttons are provided next to "Confirm": the "Reset" and "Abort" buttons. The "Reset" button will clear all the prompt area fields so that no answer or option is selected. The "Abort" button will abort the prompt, which at the current software version means that the procedure execution *will be aborted*.

GUI prompts have a programmable delay. If a prompt appears and there is no user input during a certain amount of time, the prompt area will start blinking in yellow and (optionally) a warning sound will be played. The delay time and the wav file to be played can be changed from the preference pages.

### 3.2.7 Other features

It is possible to detach a procedure view from the GUI main window in order to see two (or more) procedure views at the same time. To do this, right-click on the procedure view title and select "Detached" in the appearing pop-up menu, as shown in the Figure 12.



**Figure 12: Detaching a procedure view**

Once the view is detached it will appear as a separate window. The procedure can still be controlled from that standalone view.

To attach the view again to the main GUI window, just unselect "Detached" in the same pop-up menu of the view title tab. The standalone view will go back to its original place.

A procedure view (or any of the views of the GUI) may be maximized or restored too, by double-clicking on their title tab.

Minimization is not allowed for procedure views but for view areas, that is:

- The workspace area, containing all the procedure views and the master view
- The navigation view area
- The call stack view area

Once minimized, it is possible to restore these areas by clicking on the corresponding minimized icons, as shown in Figure 13.



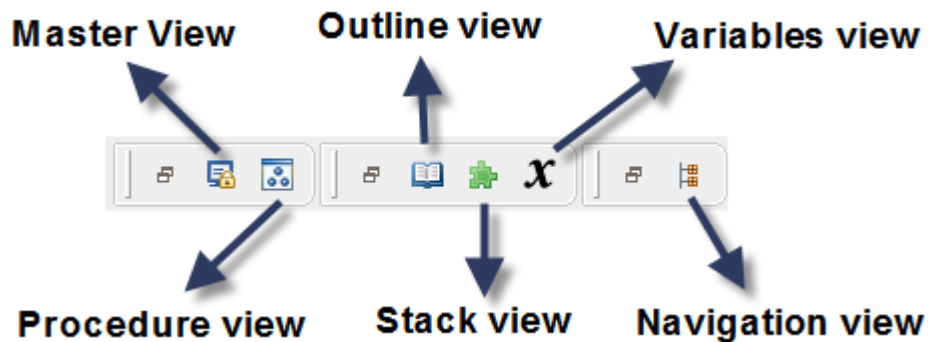


Figure 13: Restore view area icons

### 3.3 The master view

The master view is always open (is not closable) and contains two different areas:

- Master console (top part)
- Application log viewer (bottom part)

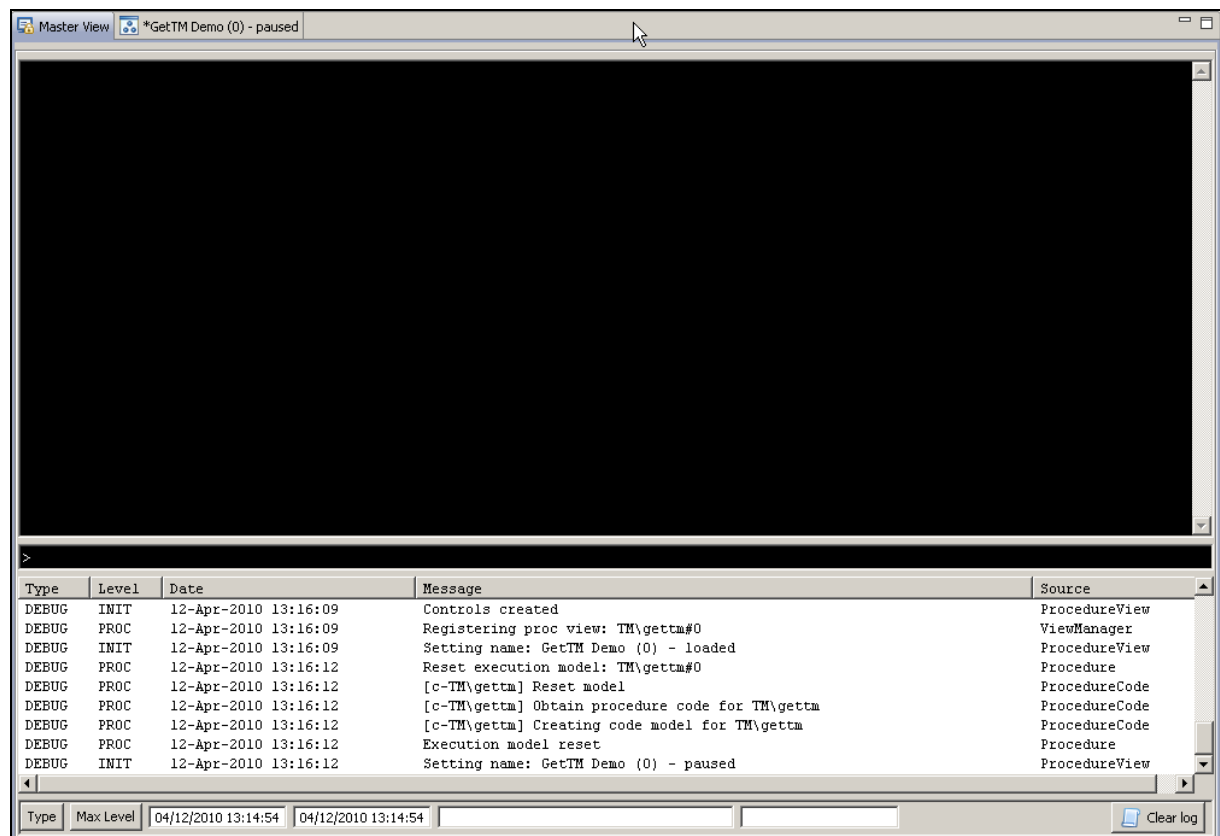


Figure 14: Master view

### 3.3.1 Master console

All the operations that can be performed with the toolbar buttons, menu commands, etc. can be carried out from the master console. The master console provides this set of commands:

- **help**: shows the list of available commands.
- **help <command>**: shows the help for a given command
- **server**: manage the connection to the SPELL server
- **attach / detach**: attach/detach the GUI to a given context
- **start / stop**: start/stop a given context
- **load / unload**: load or unload a given procedure
- **release**: release an owned procedure
- **control**: acquire the control of a given procedure
- **monitor**: monitor a given procedure
- **show**: show lists (available procedures, running procedures, etc)
- **info**: show information about items

```
>help
Available commands:
- help
- server
- attach
- start
- stop
- detach
- load
- unload
- release
- kill
- control
- monitor
- show
- info
```

The following table describes console commands in detail:

Command	Arguments	Description
server	<hostname> <port>	Connect to SPELL server at the given host and port
attach	<context name>	Attach to the given context name, once connected to a server
detach	none	Detach from the current context
start	<context name>	Start the given context process
stop	<context name>	Stop the given context process
load	<proc identifier>	Start the given procedure
unload	<proc identifier>	Unload the given procedure
release	<proc identifier>	Release the control of the given procedure
kill	<proc identifier>	Kill the given procedure
monitor	<proc identifier>	Attach to the given procedure in monitoring mode
control	<proc identifier>	Take control of the given procedure
show	contexts	Show a list of the available contexts in the SPELL server
show	procedures	Show a list of the available procedures in the current context
show	executors	Show a list of the active procedures in the current context
info	server	Show information about the current SPELL server
info	context <ctx name>	Show information about the given context
info	procedure <identifier>	Show information about the given procedure file
info	executor <identifier>	Show information about the given active procedure

### 3.3.2 Log viewer

It shows all the log messages sent by the GUI application. Messages can be filtered by type, level, data or strings.

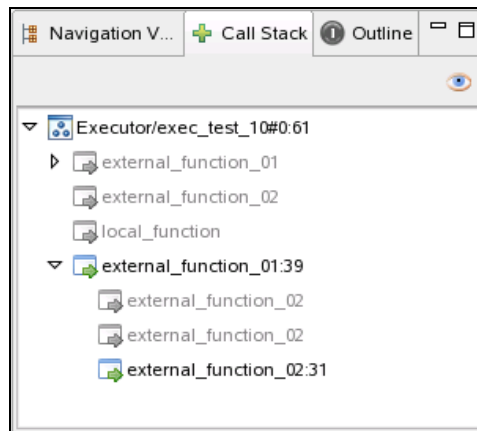
- Message type can be INFO, WARNING, ERROR and DEBUG.
- Message levels can be INIT (initialization), PROC (processing), GUI and COMM (communications)

## 3.4 The call stack view

The call stack stores information about the subroutines executed inside a procedure. Its main use is to keep track of the execution along its execution time.

In the call stack there are two kinds of subroutines, called active and inactive. The active subroutines are those which have been called but have not finished yet, while inactive ones have finished their execution.

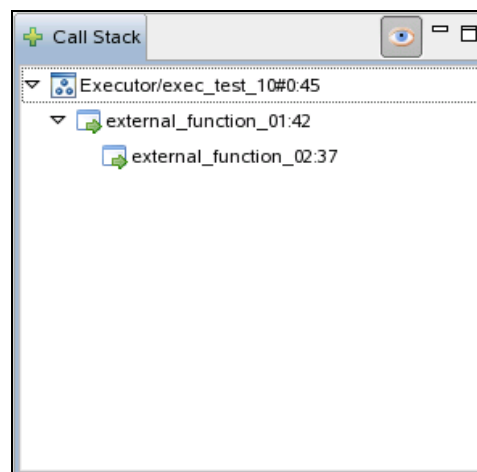
SPELL GUI provides a view in its workbench which helps to keep track of the execution procedure through the stack. In this view, subroutines are shown in a tree way, where each node represents a subroutine.



**Figure 15: Call stack view**

As shown in the image, the root node shows the procedure identifier, and represents the main routine, indicating the currently executed line. All of its children represent subroutines, which are shown with black text if they are active or gray if they are inactive. Active ones also show the on-execution line. Any node can have children, each of them representing subroutines called from their parent routine.

Inactive nodes can be hidden in the view by clicking on the eye button located at the top right. These inactive nodes will be hidden all the time the button remains toggled.



**Figure 16: Hiding inactive nodes**

From the call stack view, users can change the code being shown in the Procedure Code presentation to be the one corresponding to any of the nodes of the call stack tree. This way, it is possible to navigate through the execution history of the procedure and see the data received and the operations performed in the past. The current code shown can be changed by double-clicking on one of the items of the call stack view.

The navigation feature is only available when the procedure is in PAUSED, WAITING or ERROR states. If the procedure goes to stepping or RUNNING state again, the code view is reset to the currently executed code automatically.

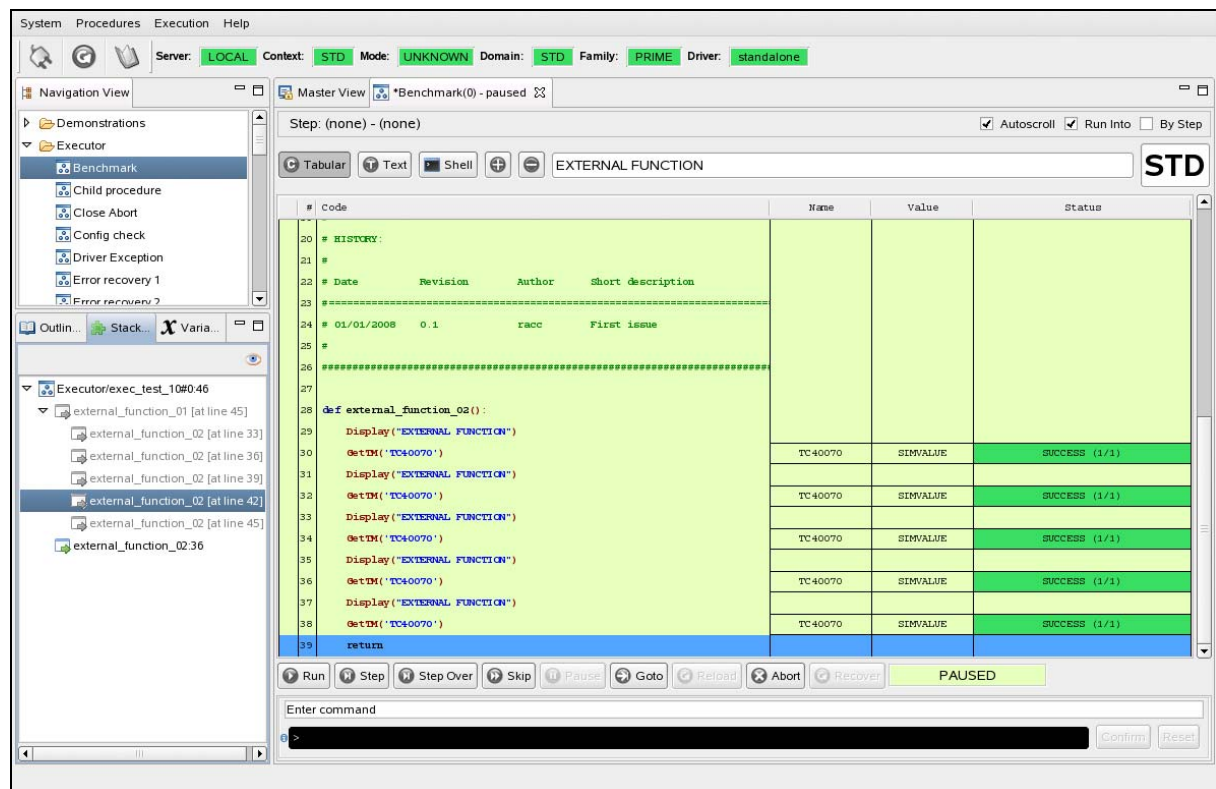


Figure 17: Clicking on a node leads to the source code line

### 3.5 The Outline view

The outline view in SPELL displays a description of the code shown in the currently focused Procedure View. The contents shown in this view depend on the source code currently displayed. These contents include source code elements like:

- Step instructions, showing all the step definitions in the procedure.
- Goto instructions, showing all the calls to the Goto function in the code.
- Function declarations, indicating all the function names and where are declared.

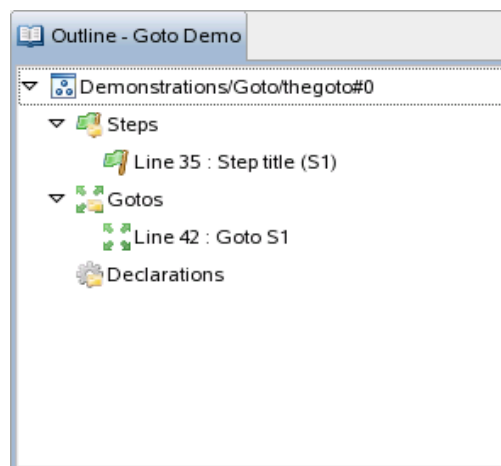


Figure 18. Outline view

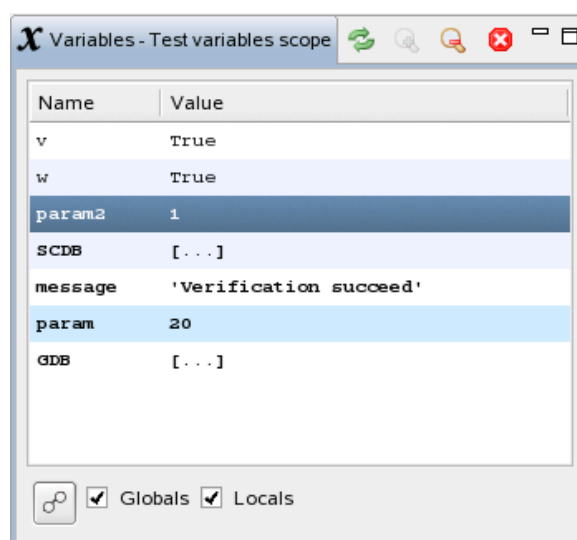
### 3.5.1 Interacting with the outline view

By double clicking on any element of the outline view, the Code Presentation will automatically move to show the corresponding element of the source code.

This feature is only enabled when the procedure execution is paused. While running, double clicking on the outline view will have no effect.

## 3.6 The Variables view

The Variables view displays information about the variables existing in the current execution scope of the currently focused procedure. During a procedure execution, users can check the existing variables and see (or modify) their values.



**Figure 19. Variables view**

In the outline view, there are two kinds of variables. Local variables are the ones which have been declared inside the current execution frame, while global variables are those which have been declared in a different scope but can be accessed in the current one. Local variables are shown in the view using normal font, while a font with a bold style is used for showing global variables.

Users can choose whether to show or hide local and global variables just by changing the selection of the buttons labelled with the text Globals and Locals.

Notice that for the sake of performance, variable changes are not automatically updated during the procedure execution. To update them, a request shall be explicitly issued as explained in the following section. There is an exception to this, which is the currently watched variables (see below).

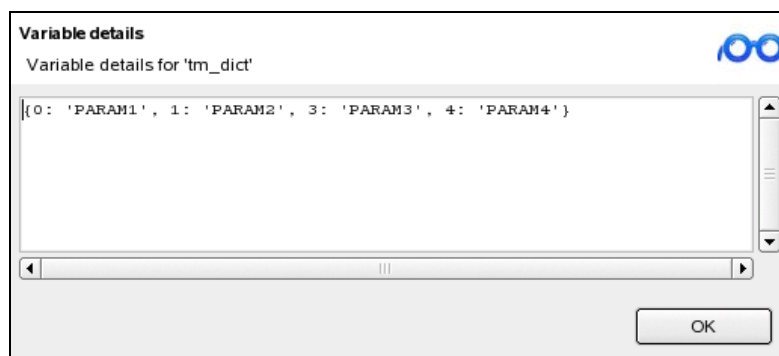
### 3.6.1 Checking variable values

During the procedure execution, users can update the variable values by clicking on the refresh button.



**Figure 20. Refresh button icon**

If a variable value to check is too long to be shown in the available area the value field will show the symbol "[...]" and users may see the details by double clicking on the variable name. A new dialog with the contents of the variable will appear.

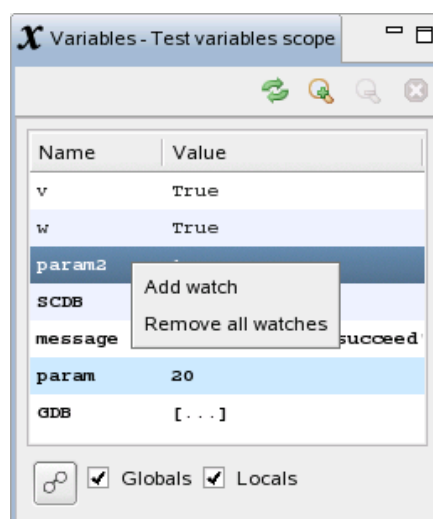


**Figure 21. Variables detail dialog**

### 3.6.2 Watch of variables

It is possible to register interest on a certain variable or set of variables in particular. This is called a "variable watch" and can be created via the variables view. Any watched variable will be monitored from the GUI, and value changes will be detected and highlighted automatically.

To create a variable watch, a variable needs to be selected on the view table. Then, the "Watch variable" button can be used to register the watch, or alternatively the "Watch variable" option of the table contextual menu which appears by right-clicking on it.



**Figure 22. Variables view context menu**

Value changes for watched variables are highlighted with a light blue background color in the table.



**Figure 23. Watch variable button icon**

Whenever a variable watch is not useful anymore, it can be destroyed or disabled by selecting the variable and clicking on the "Stop watch" button on top of the variables view. Alternatively, the corresponding contextual menu item can be used.



**Figure 24. Remove watch of variable button icon**

It is possible to disable all watches at once by using the "Stop watching variables" button on the top of the variables view.



**Figure 25. Stop watching variables button icon**

### 3.6.3 Value modification

During the procedure execution, variable values can be adjusted to perform a correct execution. There are several ways of changing the value of a variable. This also can be done directly from the variables view.

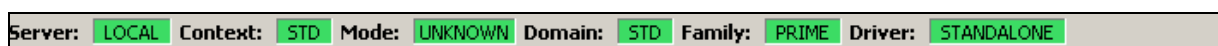
If the procedure status is paused, users can modify a value by clicking on the value field. A text editor will be shown in the place where the value was being shown. If the user modifies the value and press ENTER when the edition is finished, then its value will be updated. Pressing ESC key will cancel the modification.

It is possible to set a new value for a given variable in different ways:

- By typing the value directly. Examples are 1, 'String', [1,2,3,4], {'key' : 'value'}. It is similar as setting python variables with an explicit value.
- By calling functions. An examples is range(0,10), which creates a list containing 10 sequence elements starting from 0. If values are set this way, the value column in the variables view will show the returned value from calling the function. In the given example, the Value column will show a list containing the elements [0,1,2,3,4,5,6,7,8,9]

## 3.7 The status bar

The status bar is visible alongside the toolbar, on top of the GUI:



**Figure 26: Status bar**

It shows information about:

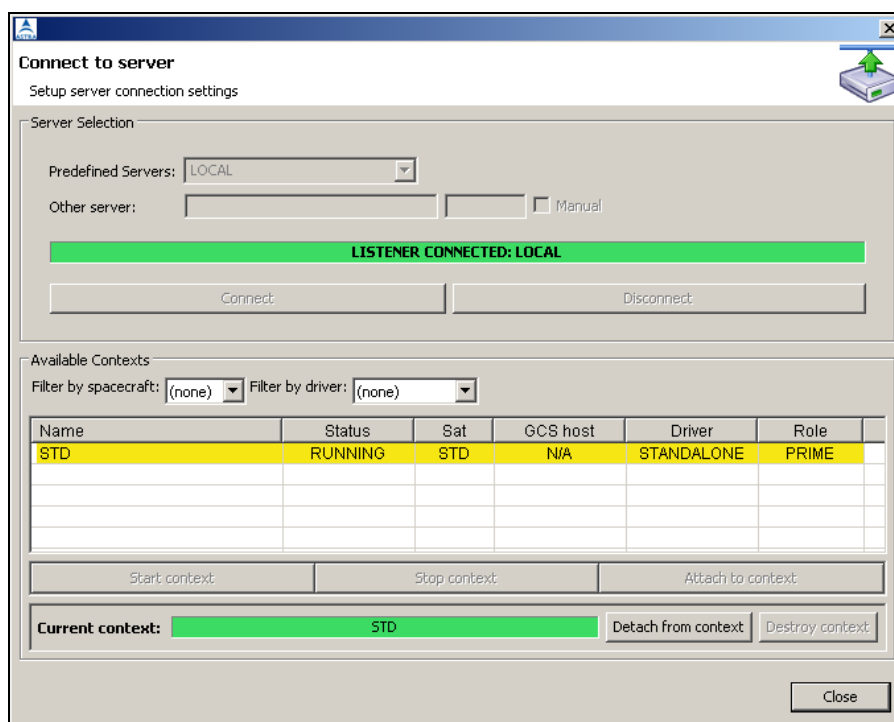
- Server: the current SPELL server being used
- Context: the current context being used
- GUI mode: commanding or monitoring
- Domain: the spacecraft name
- Family: when applicable, depending on the GCS, PRIME or BACKUP
- Driver: the name of the SPELL driver being used.

Fields appear in green background when status is nominal, yellow if they are in pending status and in red if there is a failure.

### 3.8 The connection dialog

The connection dialog is where the user manages all the GUI connections with the SPELL server. It is divided in two areas, one in relation to Listener connections and the second for Context connections.

In the listener area it is possible to select the listener to connect to from a list of predefined ones, or to manually specify the address and port of a listener. Once the listener information is selected or typed, the "Connect" button has to be used to establish the listener connection. If everything goes right, the information area just above the buttons shall become green and show the name of the concrete listener being used.



**Figure 27: Connection dialog**

As soon as the listener is connected, a list of available SPELL contexts appears. This list consists of the set of contexts managed by the listener the GUI is connected to. For each context the following data are shown:

- Context name
- Context status: AVAILABLE / RUNNING
- Corresponding spacecraft
- GCS host name
- SPELL driver
- GCS host role or family

Those Contexts which are already running will appear in bright yellow. These contexts may be attached or detached by selecting them and by using the buttons "Attach to context" and "Detach from context".

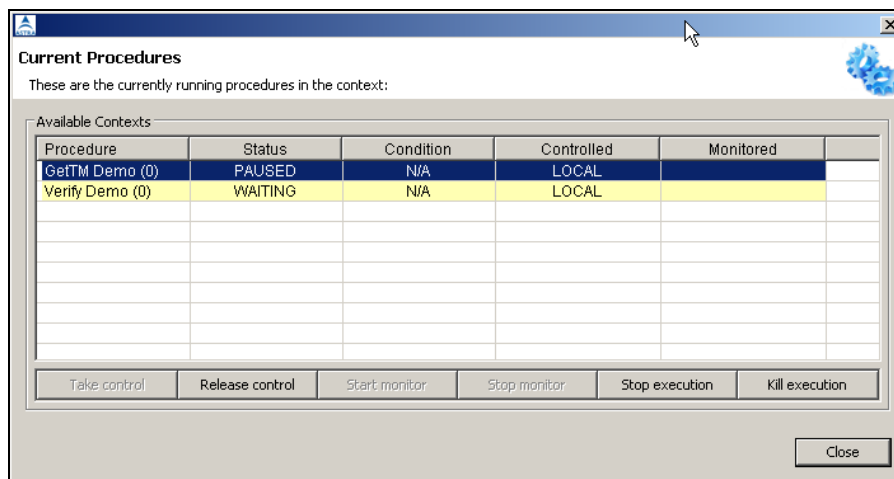
To start an available context, select it and click on "Start context" button. It is possible to stop a context as well if nobody is logged in by using the "Stop context" button. The "Current context" area should become green as soon as a valid context is attached by the GUI.

A "Destroy context" button is provided in order to kill a context process in case of hang-ups.



### 3.9 The executors dialog

The executors dialog shows information about all the procedure executions taking place in the current SPELL Context. The executions don't have to belong to the current user of the GUI, but they may be controlled by other clients working in the same context.



**Figure 28: Executors dialog**

The dialog shows a list containing:

- Executor names (with instance number)
- Status of the execution
- Client that is controlling the execution, if any
- List of clients monitoring the execution, if any.

From this dialog the following tasks can be carried out:

- Take control of an ongoing execution: provided that there is no other client controlling it already (see "Controlled" column)
- Release control of an execution. This can be done if the procedure is authorised to run in background mode.
- Start monitoring a procedure: the GUI will show execution information but it will not be possible to issue any command to the procedure or to give any user input.
- Stop monitoring a procedure: if the GUI is monitoring the execution of the given procedure, disconnect from it.
- Stop execution: provided that the GUI is the controlling client.
- Kill execution: force the procedure closure in any case, provided that the GUI is the controlled client.

It is possible to select several executors at the same time, taking into account that the enabled buttons in each case depends on common characteristics of the executors. For example, the release control button would be enabled *only* if all selected executors are being locally controlled.

## 4 Startup and configuration

### 4.1 Startup

The application is started by using launcher scripts. These scripts can be found in the <SPELL\_HOME>/bin directory. Depending on the platform where the application is being run (Windows or Linux) a different GUI launcher be available on this directory:

- SPELL-GUI for Linux platforms
- SPELL-GUI.bat for Windows platforms

Running these launchers is enough to start the SPEL-GUI with default configuration. This means that the application will be using the following configuration file:

```
<SPELL_HOME>/config/gui/config.xml
```

### 4.2 Configuration

The GUI default configuration is stored in a single XML file. This configuration specifies:

- GUI behavior at startup (auto-connection features)
- GUI appearance (colors)
- Connection settings (available SPELL servers)

These properties can be changed through the preferences management system. GUI configuration and preferences management will be explained in the following sections.

#### 4.2.1 Configuration file

The XML code below corresponds to the GUI typical configuration file:

```
<?xml version="1.0"?>
<configuration>
  <property name="AppName">SPELL</property>
  <property name="UseTraces">YES</property>
  <property name="ShowDebug">YES</property>
  <property name="DebugLevel">PROC</property>
  <property name="ConnectAtStartup">YES</property>
  <property name="InitialServer">LOCAL</property>
  <property name="InitialContext">STD</property>
  <property name="ResponseTimeout">8000</property>
  <property name="OpenTimeout">22000</property>
  <property name="ProceduresEditor"></property>
  <property name="LastServerConnected"></property>
  <property name="LastHostConnected"></property>
  <property name="LastPortConnected"></property>
  <property name="LastConnectionManual">NO</property>
  <property name="PromptSoundFile"></property>
  <property name="PromptSoundDelay">60</property>
  <property name="PreferencesEnabled">YES</property>
  <presentations>
    <presentation name="Tabular" default="yes" />
    <presentation name="Text" />
    <presentation name="Shell" />
  </presentations>
  <appearance>
    <fonts>
      <font id="MASTERC" face="Courier New" size="10" style="norm" />
      <font id="CODE" face="Courier New" size="9" style="norm" />
      <font id="TEXT" face="Courier New" size="9" style="norm" />
    </font>
  </appearance>
</configuration>
```

```

<font id="HEADER" face="Sans" size="12" style="norm" />
<font id="BANNER" face="Arial" size="25" style="bold" />
<font id="GUI_BOLD" face="Sans" size="9" style="bold" />
<font id="GUI_NOM" face="Sans" size="10" style="norm" />
</fonts>
<styles>
<style id="PROC" font="TEXT" color="0:0:0" style="norm" />
<style id="SYS" font="TEXT" color="90:90:90" style="norm" />
<style id="CFG" font="TEXT" color="0:0:0" style="norm" />
<style id="STEP" font="TEXT" color="0:0:0" style="bold" />
<style id="PROMPT" font="TEXT" color="0:0:0" style="norm" />
<style id="OTHER" font="TEXT" color="0:0:0" style="norm" />
</styles>
<colors>
<statuscolors>
<color id="SUCCESS">60:220:100</color>
<color id="WARNING">245:230:20</color>
<color id="ERROR">255:115:115</color>
<color id="FAILED">255:115:115</color>
<color id="SUPERSEDED">60:220:100</color>
<color id="IN PROGRESS">0:128:200</color>
<color id="SKIPPED">245:230:20</color>
<color id="TIMEOUT">245:230:20</color>
<color id="CANCELLED">245:230:20</color>
<color id="WAITING">245:230:20</color>
<color id="UNKNOWN">255:255:255</color>
</statuscolors>
<guicolors>
<color id="TEXTVIEW_FG">0:0:0</color>
<color id="TEXTVIEW_BG">225:235:240</color>
<color id="CONSOLE_FG">255:255:255</color>
<color id="CONSOLE_BG">0:0:0</color>
<color id="CONTEXT_ON">245:230:20</color>
<color id="CONTEXT_ERROR">255:115:115</color>
<color id="TABLE_BG">255:255:255</color>
<color id="TABLE_BG2">230:240:230</color>
<color id="ITEMS">0:0:0</color>
<color id="HIGHLIGHT">80:165:255</color>
</guicolors>
<proccolors>
<color id="UNINIT">255:255:255</color>
<color id="LOADED">235:235:235</color>
<color id="RUNNING">185:255:215</color>
<color id="WAITING">255:255:179</color>
<color id="PAUSED">225:255:185</color>
<color id="ERROR">255:185:185</color>
<color id="ABORTED">255:185:185</color>
<color id="FINISHED">220:185:255</color>
<color id="RELOADING">225:255:185</color>
<color id="INTERRUPTED">255:255:179</color>
<color id="UNKNOWN">255:255:255</color>
</proccolors>
</colors>
</appearance>
<servers>
<server>
<name>SERVER NAME</name>
<host>100.99.99.99</host>
<port>9988</port>
<role>COMMANDING</role>
</server>
<server>
<name>SERVER 2 NAME</name>
<host>100.99.99.88</host>
<port>9988</port>
<role>COMMANDING</role>
<user>spell</user>
<password>mypass</password>
</server>
</servers>
</configuration>

```

The name of the root element is “configuration”. Directly under this tag a set of “property” tags can be seen.

These properties are global options for the GUI application; they are described below:

- **AppName:** The application name to use
- **UseTraces:** Shows or hides GUI execution traces
- **ShowDebug:** enables or disables the debug traces in standard output
- **DebugLevel:** configures the traces detail
- **ConnectAtStartup:** enables or disables the autoconnection feature
- **InitialServer:** if autoconnection is used, this identifies the SPELL server that the GUI will try to connect to automatically
- **InitialContext:** if auto-connection is used, this identifies the SPELL contexts that the GUI will attempt to connect to after connecting to the SPELL server. This option may have the value "NONE". In that case, once the SPELL server is connected, the GUI does not try to connect to any context
- **ResponseTimeout:** timeout in milliseconds for the GUI-SPELL server TCP communications.
- **OpenTimeout:** timeout in milliseconds for opening and loading a procedure.
- **ProceduresEditor:** The path to an external tool used for procedures edition.
- **LastServerConnected:** The server the GUI connected on a previous session.
- **LastHostConnected:** The host the GUI connected on a previous session.
- **LastPortConnected:** The port used to connect to a host on a previous session.
- **LastConnectionManual:** Specifies if the connection during a previous session was done manually. Otherwise it was done automatically using the servers specified in this file.
- **PromptSoundFile:** The path to the sound file to use on prompts
- **PromptSoundDelay:** The time in seconds to wait before start playing the sound
- **PreferencesEnabled:** Enables or disables the preferences management system in the GUI

Properties `LastServerConnected`, `LastHostConnected`, `LastHostConnected`, `LastConnectionManual` will only contain values if the configuration file to be loaded is resulting from another GUI instance preferences export. Preferences exporting and importing mechanism is presented in section [4.2.3](#).

After the application property tags, three other main sections can be seen: "presentations" , "appearance" and "servers". The first identifies the available procedure presentation plugins, the second contains some GUI color/font configurations, and the latter specifies a list of possible SPELL servers to connect to.

#### 4.2.1.1 Presentations

A presentation is made available for the application in two steps: (1) install the corresponding RCP plugin in the SPELL GUI plugins directory, and (2) add a presentation tag in the configuration file.

The presentation tags specify the presentation name as defined in their plugins. Only one of the presentations in this section can have the *default* property set to "yes". This presentation will be the one shown by default when procedure views are open.

#### 4.2.1.2 Appearance

The appearance tag contains three subsections: (1) The fonts to use inside the application, (2) The styles to used for the scoped messages, (3) and the system colors. System colours are splitted into three groups: (1) the status colors, the ones used to represent the status of a SPELL statement being executed; (2) the GUI item colors, used in some GUI controls and views, and (3) the procedure colors, the ones used to represent the status of a SPELL procedure.

#### 4.2.1.3 Servers

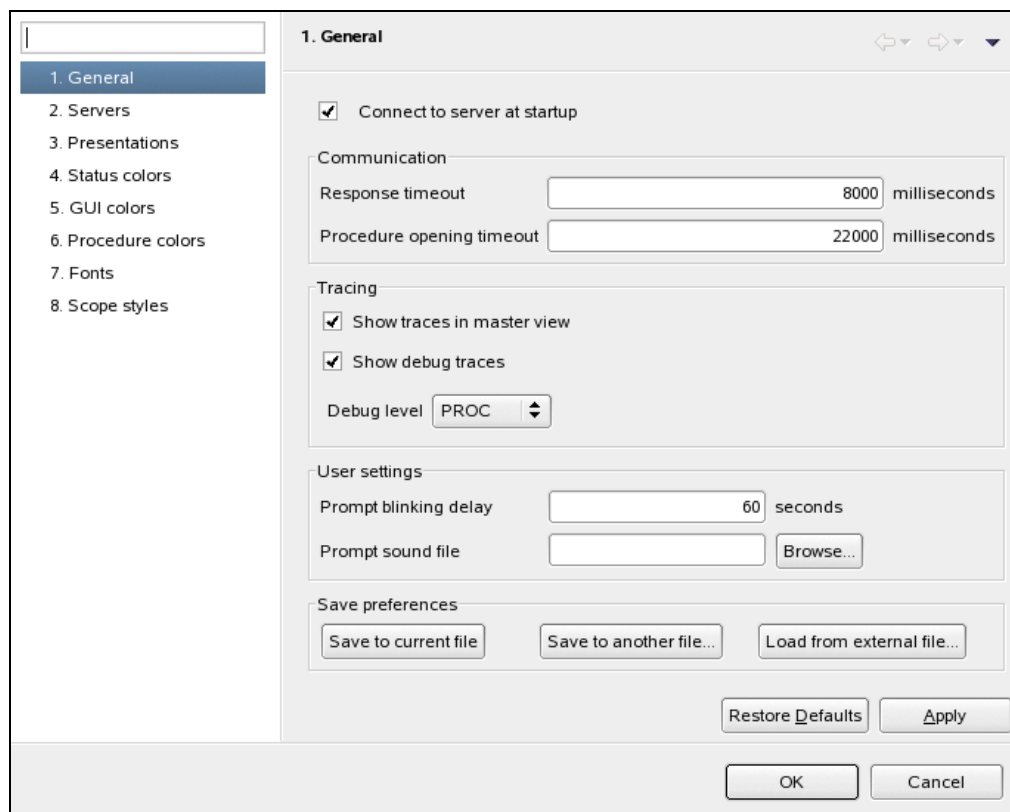
The servers tag contains a set of “`server`” items. Each server is composed of a name, a host name, role and a port number. The name field may be used in the **InitialServer** property for autoconnection purposes.

The role value can be COMMANDING or MONITORING. In the first case, the GUI connecting to this server will be able to open and monitor procedures. In the second case, the GUI will be able to monitor procedures only.

Optionally, a user name and a password may be given, if it is required to create a SSH tunnel in order to access the server. If only user name but no password is indicated, the user will be asked to provide the password at runtime.

#### 4.2.2 Preferences management

If some of the properties presented in the previous section want to be changed, this can be done by using the preferences management system. This can be accessed through the System menu at the top of the application. In that menu there is a Preferences option. By clicking on it the preferences dialog is shown.



**Figure 29. Preferences dialog**

SPELL GUI shows eight preferences dialog pages, which are shown in the dialog by making click on the left side of the dialog. Each page corresponds to one configuration file section, and allows to change its values in an easy way. At the bottom of each page, two buttons are shown to store the new values or returning to the default ones. Those buttons are labelled with **Apply**, and **Restore Defaults**.

The dialog shown at the bottom edge two more buttons. If the OK button is pressed, the values in the dialog are stored and the dialog is closed. If Cancel button is pressed, the dialog is closed without storing the new values.

The pages shown in the SPELL GUI preferences dialog are:

- **General:** Allows configuring server communication settings, tracing and prompting sound.
- **Servers:** Allows editing configuration files servers, as well as adding new ones.
- **Presentations:** Allows enabling or disabling presentation, as well as setting the default presentation
- **Status colors:** Users may change status colors in this page.
- **GUI colors:** Users may change GUI colors in this page.
- **Procedure colors:** User can change Procedure status colors in this page.
- **Fonts:** Application fonts can be changed in this page.
- **Scope styles:** Scope message styles can be changed in this page.

#### 4.2.3 Exporting and importing preferences

SPELL GUI allows to export its configuration to an external file as well as to import a configuration file from another SPELL GUI instance. This can be done through the preferences dialog.

In the General preferences page, there is a section at the page bottom which allows to perform these actions through three different buttons:

- **Save to current file:** File used for loading preferences will be overwritten with current preference values.
- **Save to external file:** Current preference values are stored in a file specified in the file dialog shown when this button is shown. If the selected file already exists, user is prompted to overwrite that file.
- **Load from external file:** Load a file selected in a dialog shown after pressing the button. When selected, current preferences are overwritten with the values stored in the file.

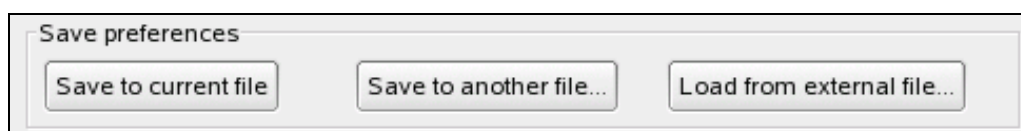


Figure 30. Import/Export preferences section

### 4.3 Initial steps

The following steps shall be carried out in order to prepare a SPELL execution session:

- Open the connection dialog
- Connect to a listener
- Select and start the desired context, if it is not running already
- Select and attach to the context
- Close the connection dialog
- Load procedures

## 5 Procedure Execution

### 5.1 Procedure execution status

Each procedure is run in what is called an execution session or *executor*. The executor is a process running in the SEE host, and is controlled by the SPELL context process. The status of the executor (that is, the status of the procedure execution) change along the lifecycle of a procedure.

The following table shows the list of possible procedure status with their description. The “Default color” column shows the color which is used in the GUI code and textual pages of a procedure view to indicate the current procedure status.

<b>Status</b>	<b>Description</b>	<b>Default Color</b>
UNINIT	<i>Unknown state</i>	
LOADED	<i>Procedure code loaded</i>	
RUNNING	<i>Executing in play mode</i>	
WAITING	<i>Execution waiting for event</i>	
PAUSED	<i>Execution paused</i>	
ERROR	<i>Load, system or syntax error</i>	
ABORTED	<i>Aborted by user</i>	
FINISHED	<i>Execution finished successfully</i>	
RELOADING	<i>Procedure loading is ongoing</i>	
INTERRUPTED	<i>Procedure has been paused while waiting</i>	
UNKNOWN	<i>Unknown status</i>	

### 5.2 Foreground and background procedures

Most procedures are executed in foreground. This means that there is always one SPELL client controlling and watching the procedure execution. However, it is possible to authorise a procedure to run in background mode.

A procedure running in background mode does not require a GUI controlling it. All procedure messages will be injected directly in the controlling system (e.g. GCS), provided that this type of communication is possible (SPELL event service shall be implemented by the driver).

If there is user input required, a background procedure raises a warning event and the execution is immediately put on hold.

A background procedure may become a foreground procedure when a SPELL client takes control of the execution.

## 5.3 Procedure execution control

Procedure execution is controlled by means of the control area buttons.

The following actions or commands are available from this area:

- **Run:** execute the procedure in play mode
- **Step:** also known as step-into. Execute the next procedure statement. If this statement calls a function provided by a subprocedure, the procedure code will be changed by the subprocedure one. After finishing the function the main procedure code will be shown again. Obviously this is applicable only to the Code Page.
- **Step Over:** same as step, but in this case no subprocedure/function code is shown, the code of the main procedure remains visible.
- **Skip:** skip the next statement.
- **Pause:** pause the procedure execution.
- **Goto:** goto to a given Step in the procedure.
- **Reload:** reload the procedure.
- **Abort:** abort the procedure execution.
- **Recover:** recover the procedure from an anormal state.

Notice that not all the buttons are clickable at all times, since not all of them can be issued always. This depends on the current procedure status. The following table shows the command availability versus the execution status:

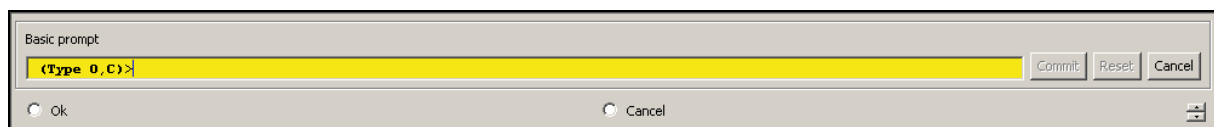
Command	RUNNING	PAUSED	WAITING	ABORTED	FINISHED	ERROR
Run		X				
Step		X				
Step Over		X				
Skip		X				
Goto		X				
Pause	X		X			
Reload				X	X	
Abort	X	X	X			
Recover						X(*)

(\*) Only if the error is not fatal (e.g. an executor process crash or a syntax error)

## 5.4 Procedure input

As it has been explained, the control area can be used to provide inputs to the procedure on demand.

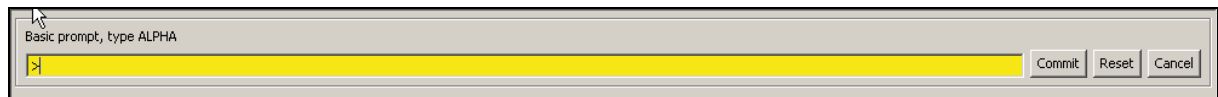
Whenever the `Prompt()` function is used in a procedure, the prompt is issued to the user in this input area. The figures below show examples of the aspect of the input area when a selection or text prompt is being executed.



**Figure 31: Input area with selection prompt**

Selection prompts consist on a list of choices for the user. The options are shown using radio buttons under the text field of the input area (a scroll bar appears when necessary), although the answer can be written as well with the keyboard. For this matter, the text field shows a hint regarding the expected answers. In the example image this hint is "O,C" which means that only those letters will be accepted.





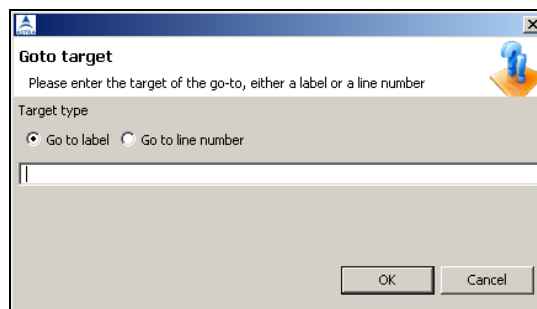
**Figure 32: Input area with text prompt**

Text prompts may accept any alphanumerical input, so that the text field is the only control shown. Unless the procedure developer specifies the opposite, there is no answer validation. For numerical prompts, the answer put in the text field shall be a decimal number.

Notice that three buttons are available in the input area: "Commit", "Reset" and "Abort". The first one will send the user input to the procedure. "Reset" button will clear all the possible selections or already typed text. "Abort" button will cancel the prompt operation, but it has to be kept in mind that *cancelling a prompt implies an immediate execution abort*.

## 5.5 Manual goto

When the procedure is in PAUSED state, the current execution line can be moved with the manual goto mechanism. This is done by clicking on the Goto control button in the procedure view. A Goto dialog will appear:



**Figure 33: Goto dialog**

Target line can be identified by a label or a line number. Valid labels correspond to the **Step** SPELL statements appearing in the procedure source code.

Due to Python scope rules, it is not possible to perform a go-to jump to an arbitrary line, though. If the current line is located within the scope of a function, the go-to target shall remain within the scope of that function. The same applies when the current line is outside a function (e.g. in the main procedure code): the go-to target shall remain within the original scope, that is, outside any function. Put another way, go-to jumps are not possible between different functions, from main code into a function, or from one function to the main code. Jumps between different procedures are also invalid.

## 5.6 Monitoring and control

The GUI from where a procedure is loaded has the control of the procedure, that is, it is able to send commands to it and control the execution.

Other GUIs may supervise the procedure execution, being able to see what is happening in the procedure, but they cannot send commands to it. It is said that these GUIs are *monitoring* the procedure.

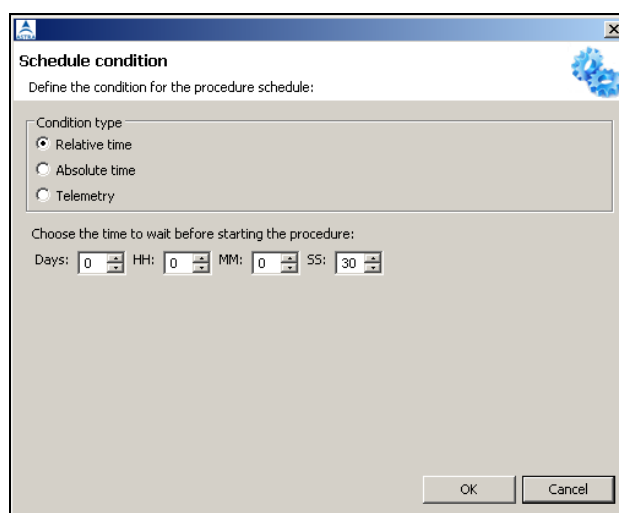
In order to monitor a procedure, the user shall select it on the Executors Dialog and click on "Start monitor" button. The procedure view will appear on the GUI, but the control area will be disabled. To finish a monitoring session, the view may be closed by selecting "Stop monitor" in the Executors Dialog.

### 5.6.1 User handover

A monitoring GUI can gain the control of a procedure if the controlling GUI releases the procedure. The procedure is released by using the Executors Dialog and clicking on "Release control" button. Once the procedure control is released, another GUI may select the procedure and gain control over it by clicking on "Take control" button.

## 5.7 Scheduling procedures

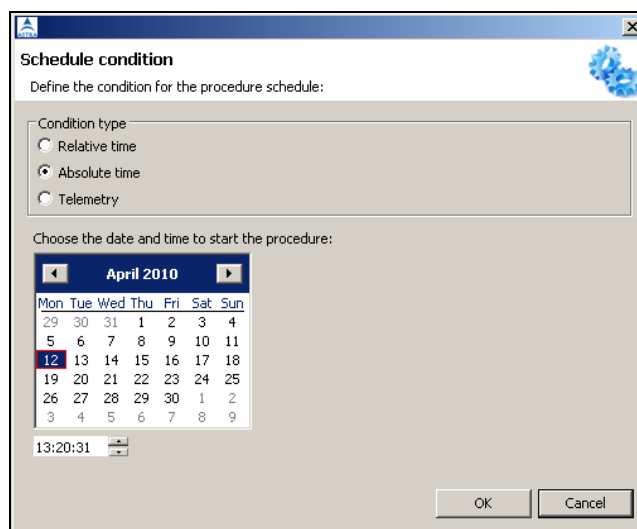
It is possible to open procedures in scheduling mode instead of opening them for immediate execution. To do so, the desired procedure has to be selected on the navigation view first. Then the menu command *Procedures/Schedule selected procedure* can be executed. A procedure scheduling configuration dialog will appear:



**Figure 34: Schedule condition dialog**

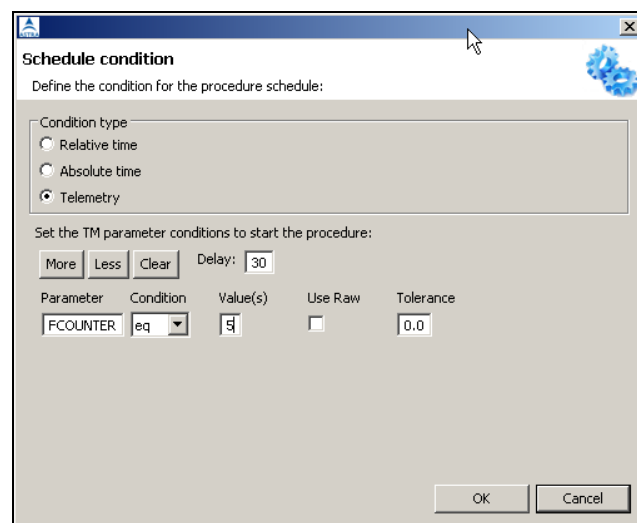
There are three possibilities for scheduling a procedure: wait a relative time before running; wait until an absolute time before running; and wait for a set of telemetry conditions before running. By default, a relative time condition is selected.

The relative time condition is specified in the form of the amount of days, hours, etc. to wait before triggering the procedure execution. The absolute time condition requires selecting a particular date and time:



**Figure 35: Scheduling with absolute time condition**

On the other hand, the telemetry condition allows selecting a set of TM points and indicating the conditions that must be satisfied before triggering the procedure execution:



**Figure 36: Telemetry scheduling condition**

The data provided for telemetry conditions are analog to the data provided to the WaitFor SPELL language function (please refer to the language reference for details).

Once the condition is specified, the procedure view appears and the procedure remains in WAITING state, until the scheduling condition is satisfied. Then the procedure goes to RUNNING mode automatically.

## 5.8 Procedure files

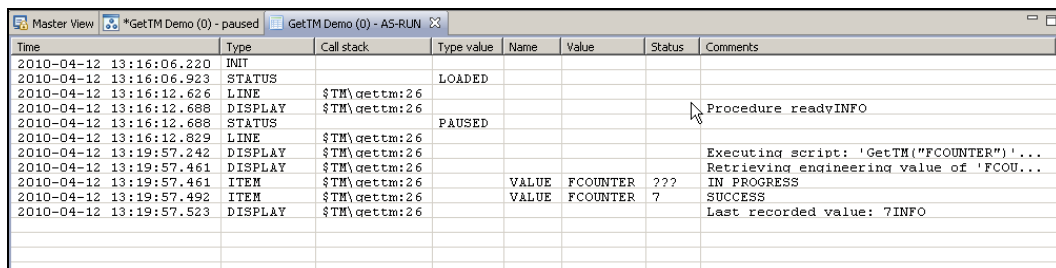
Procedure files are generated during the execution on the SPELL server side.

### 5.8.1 As-Run files

The As-Run file provides an operational log of the procedure execution. It registers all the operations being carried out by the procedure during the execution. Telemetry value acquisitions, command executions, prompt answers, all these data are logged into the As-Run file.

The As-Run file can be inspected from the SPELL GUI by selecting the desired procedure view in the GUI, and then executing the menu command *Procedures/View procedure As-Run file*. A new view will appear next to the procedure view, showing the contents of the As-Run file.

This view is static, meaning that if the procedure execution continues meanwhile in this view, no new data will be refreshed. It will be refreshed, though, every time the As-Run view is deselected and selected again (e.g. the procedure view tab is selected, then the As-Run view tab is selected again).



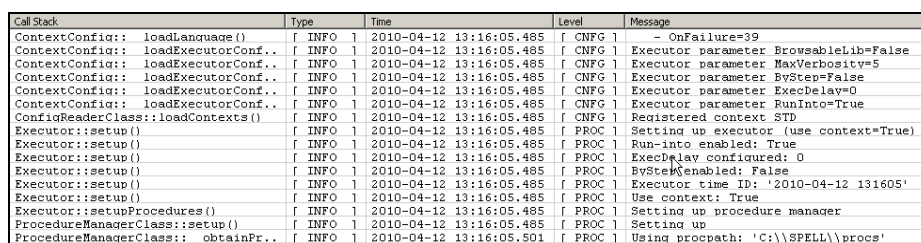
Time	Type	Call stack	Type value	Name	Value	Status	Comments
2010-04-12 13:16:06.220	INIT						
2010-04-12 13:16:06.923	STATUS		LOADED				
2010-04-12 13:16:12.626	LINE	\$TM\gettm:26					
2010-04-12 13:16:12.688	DISPLAY	\$TM\gettm:26					Procedure ready
2010-04-12 13:16:12.688	STATUS		PAUSED				INFO
2010-04-12 13:16:12.829	LINE	\$TM\gettm:26					
2010-04-12 13:19:57.242	DISPLAY	\$TM\gettm:26					Executing script: 'GetTM('FCOUNTER')'...
2010-04-12 13:19:57.461	DISPLAY	\$TM\gettm:26					Retrieving engineering value of 'FCOU...
2010-04-12 13:19:57.461	ITEM	\$TM\gettm:26		VALUE	FCOUNTER	???	IN PROGRESS
2010-04-12 13:19:57.492	ITEM	\$TM\gettm:26		VALUE	FCOUNTER	7	SUCCESS
2010-04-12 13:19:57.523	DISPLAY	\$TM\gettm:26					Last recorded value: 7
							INFO

Figure 37: As-Run view

### 5.8.2 Log files

Procedure log file provides a development-point of view log of the procedure execution. It does not register the procedure operations in a clean, summarized way as the As-Run file. Procedure log files are intended to be used for software support and debugging.

The procedure log file can be inspected from the SPELL GUI by selecting the desired procedure view in the GUI, and then executing the menu command *Procedures/View procedure log file*. A new view will appear next to the procedure view, showing the contents of the log file.



Call Stack	Type	Time	Level	Message
ContextConfig:: loadLanguage()	f INFO	2010-04-12 13:16:05.485	f CNFG	- OnFailure=39
ContextConfig:: loadExecutorConf..	f INFO	2010-04-12 13:16:05.485	f CNFG	Executor parameter BrowseableLib=False
ContextConfig:: loadExecutorConf..	f INFO	2010-04-12 13:16:05.485	f CNFG	Executor parameter MaxVerbosity=5
ContextConfig:: loadExecutorConf..	f INFO	2010-04-12 13:16:05.485	f CNFG	Executor parameter ByStep=False
ContextConfig:: loadExecutorConf..	f INFO	2010-04-12 13:16:05.485	f CNFG	Executor parameter ExecDelay=0
ContextConfig:: loadExecutorConf..	f INFO	2010-04-12 13:16:05.485	f CNFG	Executor parameter RunInto=True
ConfigReaderClass::loadContexts()	f INFO	2010-04-12 13:16:05.485	f CNFG	Registered context STD
Executor::setup()	f INFO	2010-04-12 13:16:05.485	f PROC	Setting up executor (use context=True)
Executor::setup()	f INFO	2010-04-12 13:16:05.485	f PROC	Run-into enabled: True
Executor::setup()	f INFO	2010-04-12 13:16:05.485	f PROC	ExecDelay configured: 0
Executor::setup()	f INFO	2010-04-12 13:16:05.485	f PROC	ByStep(enabled): False
Executor::setup()	f INFO	2010-04-12 13:16:05.485	f PROC	Executor time ID: '2010-04-12 131605'
Executor::setupProcedures()	f INFO	2010-04-12 13:16:05.485	f PROC	Use context: True
ProcedureManagerClass::setup()	f INFO	2010-04-12 13:16:05.485	f PROC	Setting up procedure manager
ProcedureManagerClass:: obtainPr..	f INFO	2010-04-12 13:16:05.501	f PROC	Setting up
				Using procpath: 'C:\SPELL\procs'

Figure 38: Procedure log file view

This view is static, meaning that if the procedure execution continues meanwhile in this view, no new data will be refreshed. It will be refreshed, though, every time the log view is deselected and selected again (e.g. the procedure view tab is selected, then the log view tab is selected again).

## 5.9 BREAKPOINTS

This feature is linked to, and only works with the Code Presentation.

The breakpoint mechanism allows setting automatic stop points in the procedure source code. When a breakpoint is set on a particular line and the execution reaches it, the procedure goes automatically to PAUSED state.

As described in the next sections, there are two different types of breakpoints: permanent and temporary breakpoints. Permanent breakpoints are those that remain active when the execution reaches them. Temporary breakpoints disappear when the execution pauses on them.

Only permanent breakpoints can be explicitly set by the user.

### 5.9.1 Adding or removing permanent breakpoints

Breakpoints can be added or removed from the contextual menu of the Code Presentation. The contextual menu shows different options including three breakpoint operations:

- Add/Remove a breakpoint: Set a breakpoint at the selected line
- Remove all breakpoints: Remove all breakpoints in a procedure

To add a breakpoint, select the “Add breakpoint at this line” option. After that, a red bullet at the beginning of the line will be shown. To remove the breakpoint, select the “Remove breakpoint” option from the same menu.

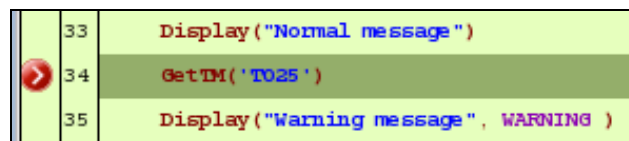


Figure 39: Line with a breakpoint

There is also another option for removing all the existing breakpoints in the source code. To remove all breakpoints, select the “Remove all breakpoints option”.

Users must take in care that whenever a procedure is reloaded, every existing breakpoint is removed automatically.

### 5.9.2 Executing a procedure until a line is reached (temporary breakpoints)

It is possible to run a procedure until a selected line and then stop. To do this, from the Code view context menu, the “Run until this line” option shall be selected. A yellow bullet (indicating a temporary breakpoint) will be shown at the beginning of the line. Then the procedure will start running, until the line with temporary breakpoint is reached; at this point the execution is stopped and the temporary breakpoint will disappear.

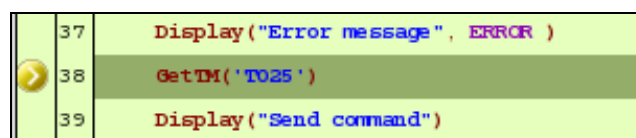


Figure 40: Execution will progress until line 38

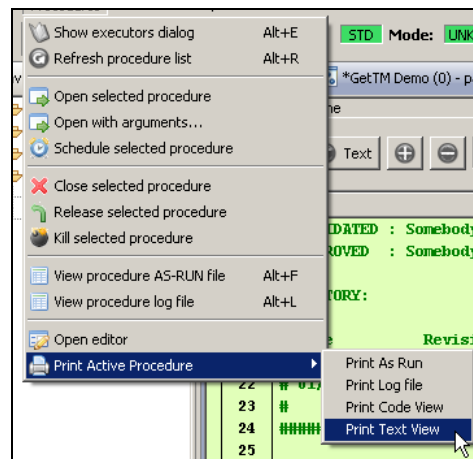
## 6 Other features

### 6.1 Printing

Several printing options are available:

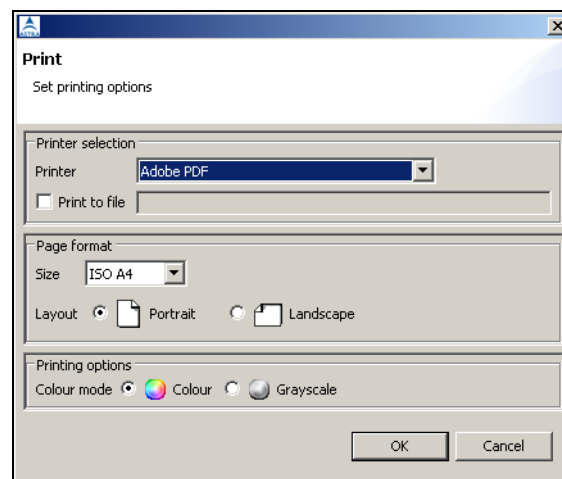
- Printing the SPELL GUI application log: via the menu command *System/Print master log*.
- Printing procedure As-Run file
- Printing the procedure log file
- Printing the procedure Tabular (or code) view
- Printing the procedure Text view

Except for the first operation, all printing commands are available only when a procedure view is selected. Once a procedure is selected, the Procedures menu changes and a new item appears at the bottom, *Print Active Procedure*:



**Figure 41: Printing procedure data**

The last four operations mentioned can be performed with these menu commands. When printing any of these data, the print dialog appears:



**Figure 42: Printing dialog**

## 6.2 Code search

When a procedure view is showing the source code through the code view, an option for searching text in the code is available. Making right click over the source code shows a context menu where different options are available. If the Search option is selected, a dialog for entering the text to search becomes visible.



**Figure 43: Text search dialog**

When the Find button in the Search dialog is pressed, all the text occurrences in the source code are highlighted, and the line which contains the first of them is selected automatically. Next and previous occurrences can also be selected by using the Find previous and Find next buttons.



**Figure 44: Search results are highlighted**

The text search dialog can be closed in two ways, depending if the search results want to be kept or not. If the Clear search close option in the dialog is selected, the dialog is closed and the highlighted texts in the code are shown their normal way. If the dialog is just closed, the found texts remain highlighted. To remove the highlighting, users can select the Clear search close option, or selected the Clear search option in the context menu.

## 6.3 Copy source code

In the code view, users can selected a source code line and copy it to make it available to paste in an external editor. To do this, a copy options is shown in the code view context menu. Once this option is selected, the source line can be pasted on any editor.

## 7 Getting help

SPELL GUI has an option on its top menu called Help, which allows checking the SPELL GUI version on use, the current release information, and give access to the documentation.

When selecting this menu, four options are shown:

- **About:** Like in any Eclipse RCP-based application, the About section shows SPELL GUI installation information. This low-level information reports about which components are installed, and the configuration parameters which are being used. It also gives access to the log file.
- **Release information:** When this option is selected, a dialog is shown describing the new features added in the on use SPELL GUI instance.
- **SPELL Language manual:** This option launches an external PDF document viewer, showing the SPELL language document, which describes all the SPELL functions and their behaviour.
- **SPELL GUI manual:** This option launches an external PDF document viewer, showing this document.