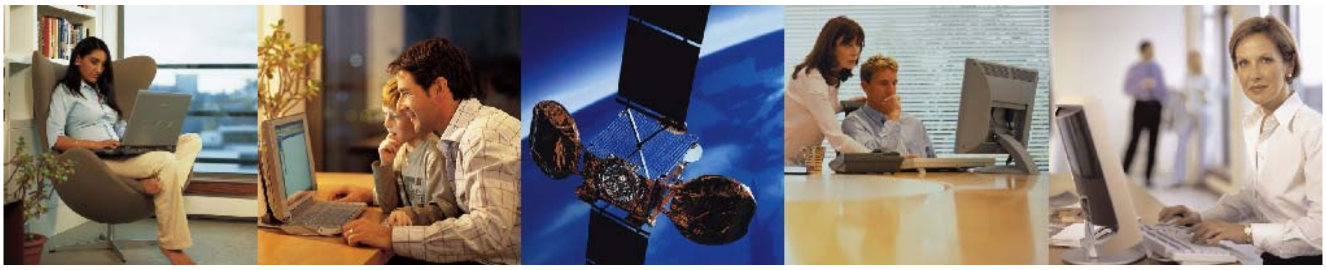
SES ENGINEERING



SPELL GUI Manual

Software version 2.0

Author	Date	Version	Comment
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
Fabien Bouleau	9 March 2011	3.5	Reviewed
Rafael Chinchilla	25 March 2011	3.5	Fixed styles, images, index

Prepared By	Rafael Chinchilla Fabien Bouleau J. Andrés Pizarro	Signature: *signature on file*
Reviewed By	Thomas Nowak	Signature: *signature on file*
Authorized By	Thomas Nowak	Signature: *signature on file*

Table of Contents

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

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 automated 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 in 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 real-time elements needed for executing procedures. It is composed of:

(a) the *SPELL server*, core of the environment, in charge of executing the procedures. It coordinates all tasks, sets up the binding with the ground control system, 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 diagram hereafter summarizes the overall structure of the SPELL:

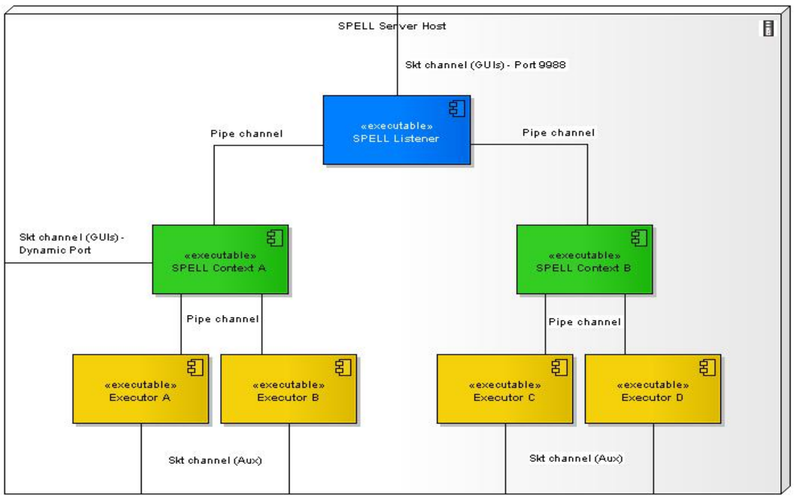


Figure 1: Structure of the SPELL server

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 the GUI with the list of all available *SPELL contexts*. A context is a dedicated process (colored in green in the picture) in charge of controlling and coordinating the execution of SPELL procedures for a specific spacecraft. Each context encapsulates data, communication and control for that spacecraft only. The 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 current status (running or not) of each of the context is indicated in the list displayed in the GUI. t 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. A direct connection is then established between the GUI and the context. At this point, the GUI can start working with SPELL procedures.

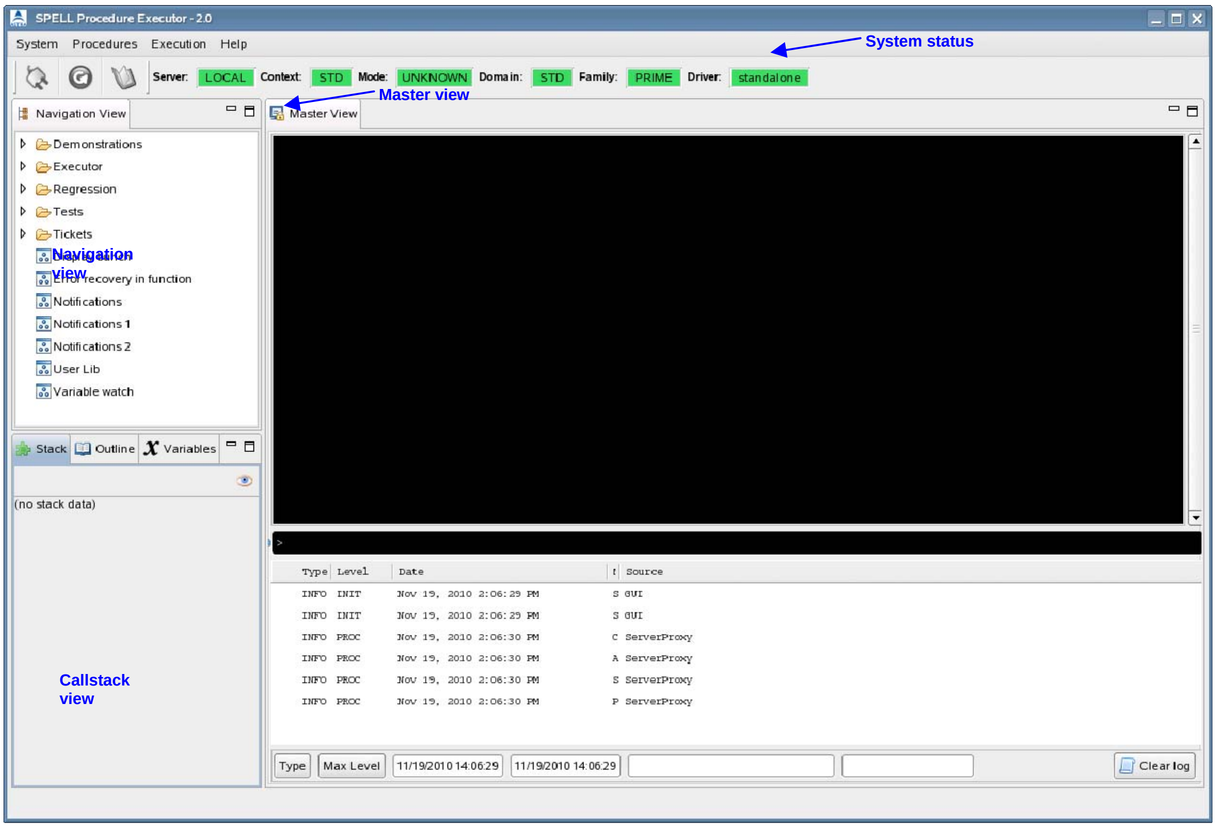
3 SPELL GUI components

The SPELL GUI is an Eclipse RCP-based application. Its design therefore follows the Eclipse IDE guidelines and concepts, and its appearance is identical to the 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. The GUI also provides a menu and a toolbar, as well as some dialogs for feature configuration or for performing contextual tasks.

The GUI is composed of the following views:

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

The following screenshot represents a general view of the SPELL GUI. Note that in this screenshot, no procedure has been open yet; no procedure view is therefore displayed. Also notice both Navigation view and call stack view are both stacked in the navigation area.



Server: LOCAL Context: STD Mode: UNKNOWN Domain: STD Family: PRIME Driver: standalone

(no stack data)

```
Type Level  Date                          ! Source
INFO INIT   Nov 19, 2010 2:06:29 PM       S GUI
INFO INIT   Nov 19, 2010 2:06:29 PM       S GUI
INFO PROC   Nov 19, 2010 2:06:30 PM       C ServerProxy
INFO PROC   Nov 19, 2010 2:06:30 PM       A ServerProxy
INFO PROC   Nov 19, 2010 2:06:30 PM       S ServerProxy
INFO PROC   Nov 19, 2010 2:06:30 PM       P ServerProxy
```

11/19/2010 14:06:29 11/19/2010 14:06:29

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 organized in folders. The folder structure reflects the directory structure of the procedure base directory located on the SPELL server host. This directory is specified in the SPELL context configuration file (its location is context-specific).

The screenshot hereafter shows an example of navigation view.

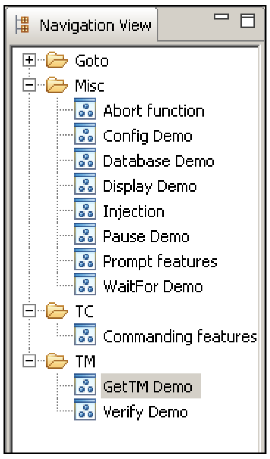
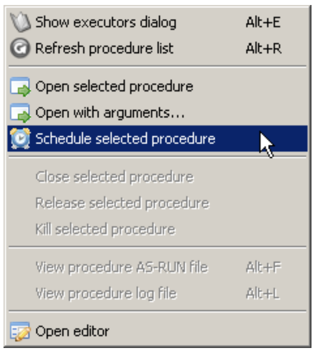


Figure 3: Navigation view

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 the default settings.

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

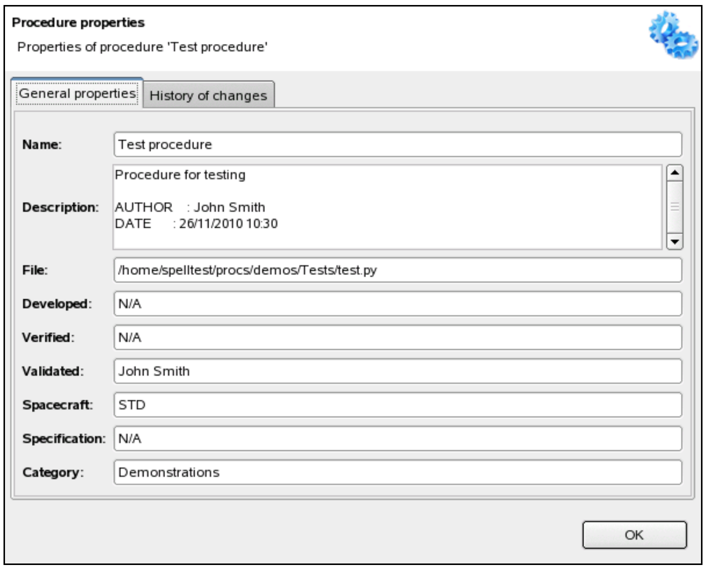

Show executors dialog Alt+E
Refresh procedure list Alt+R
Open selected procedure
Open with arguments...
Schedule selected procedure
Close selected procedure
Release selected procedure
Kill selected procedure
View procedure AS-RUN file Alt+F
View procedure log file Alt+L
Open editor

- *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 scheduling mode. The user may then specify a time or telemetry condition for the schedule.

Whenever procedure files are created or modified on the server, the "*Refresh procedure list*" button or menu command (in the *Procedures* menu) will refresh it in the GUI by issuing a request to the context.

3.1.1 Procedure properties

Procedure source files define properties in the header section, such as the author, the versioning information or the change history. This information can be accessed by right-clicking on the procedure title on the Navigation View, and selecting "Procedure properties". The procedure information will then appear in a popup dialog.


Procedure properties
Properties of procedure 'Test procedure'

Name:	Test procedure
Description:	Procedure for testing AUTHOR : John Smith DATE : 26/11/2010 10:30
File:	/home/spelltest/procs/demos/Tests/test.py
Developed:	N/A
Verified:	N/A
Validated:	John Smith
Spacecraft:	STD
Specification:	N/A
Category:	Demonstrations

Figure 4. Procedure properties dialog

The General properties tab of the dialog shows general information about the procedure, such as the author, description or target spacecraft this procedure was created for, 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 one of the application. It allows the user to watch what is going on during the execution of the procedure. It also provides means for controlling the execution, provided the GUI is in control.

Multiple procedure views can be concurrently open. Each view is linked to a single procedure instance. Note that it is possible to open several instances of the same procedure within a given SPELL Context; each view is linked to one of the instances. Each instance is identified by the procedure identifier plus an instance number, starting at zero.

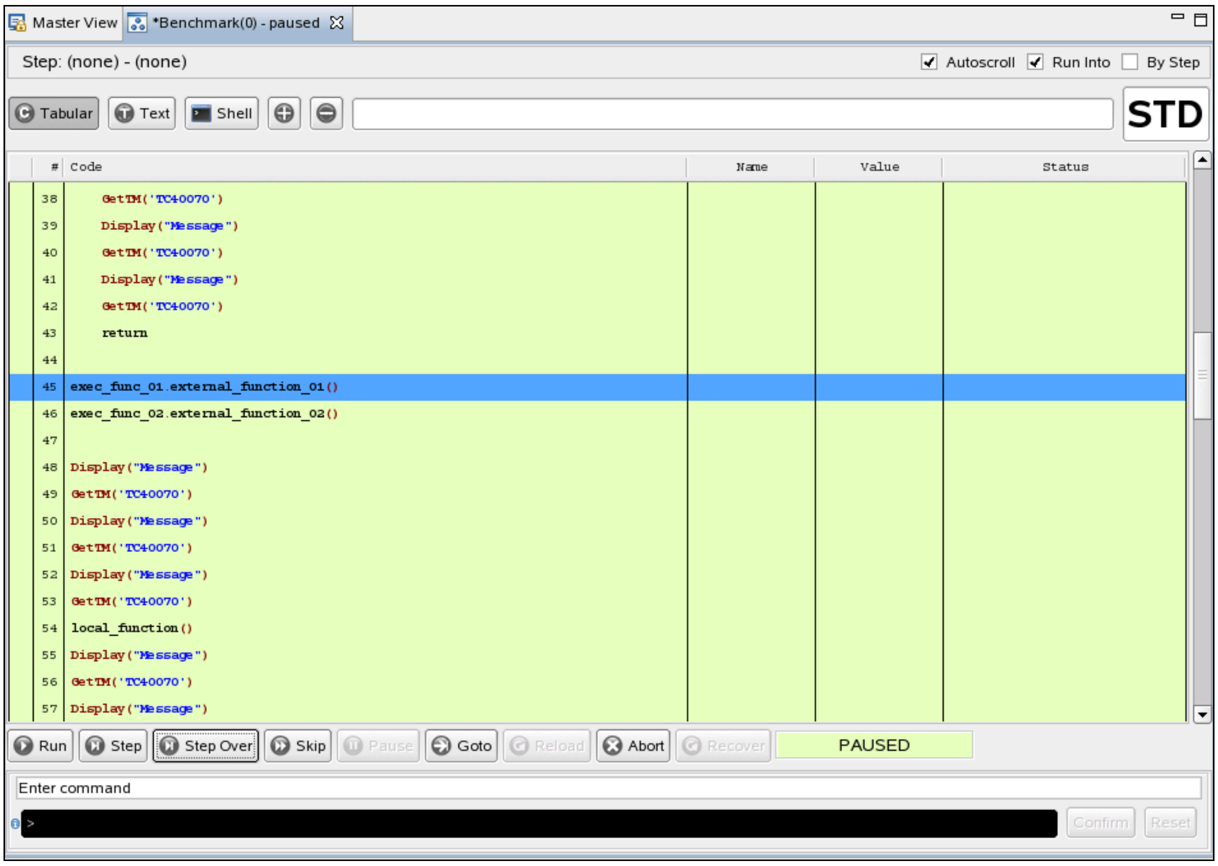
All procedure views are arranged on the GUI workspace area (initially showing the Master view only), and they are selectable by clicking on their corresponding title tab.

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

The *Tabular*, or code presentation, presents the SPELL procedure source code and beside of which is displayed a set of three columns for item names, values and status. These columns are filled with the telemetry mnemonic and value whenever a telemetry point is acquired.

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

The following screenshot shows an example of procedure view with the *Tabular* presentation



```
    GetTM('TC40070')
    Display("Message")
    GetTM('TC40070')
    Display("Message")
    GetTM('TC40070')
    return

exec_func_01.external_function_01()
exec_func_02.external_function_02()

Display("Message")
GetTM('TC40070')
Display("Message")
GetTM('TC40070')
Display("Message")
GetTM('TC40070')
local_function()
Display("Message")
GetTM('TC40070')
Display("Message")
```

Figure 5: Procedure view example

The main areas of this view are:

- **Presentations area:** contains the presentation selection buttons (depending on the ones that are available), the zoom controls, the main message display and the spacecraft indicator. In the example above, the Tabular, Text and Shell presentations are installed.

- **Procedure area:** in the example above, shows the procedure code and/or execution information. The contents depends on the selected presentation).

- **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 getting user input or for executing statements in the scope of the procedure .

Note that the view title contains the procedure name, the instance number (between parentheses), and the procedure status.

3.2.1 The Tabular presentation

As previously mentioned, the Tabular or code presentation shows the SPELL procedure source code and a set of three data columns.

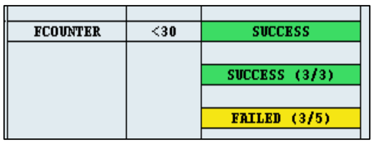
Source code lines are numbered. At runtime they are marked as they are executed to show the execution coverage.. In parallel, the currently executed line is highlighted in bright blue color.

The SPELL language syntax is highlighted in order to improve readability. The three columns on the right side of the lines of code are used to show extra information about the execution of the SPELL statement, if relevant.

This information normally consists in:

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

The information actually displayed in these columns depends on the statement being executed: for complex statements that contain several items (i.e. a multiple TM verification), the information displayed will consist in the 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. The item name and item value columns will be left blank.

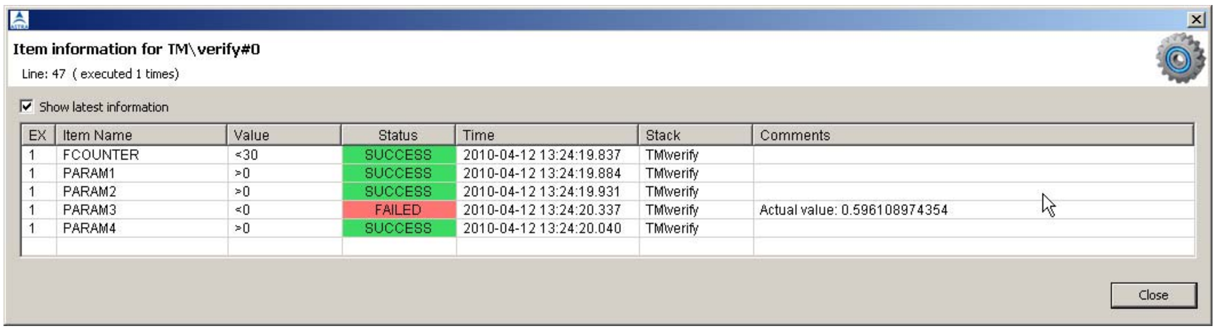


FCOUNTER	<30	SUCCESS
		SUCCESS (3/3)
		FAILED (3/5)

Referring to the opposite screenshot, three procedure lines produced item information:

- The first statement successfully checked that the TM point 'FCOUNTER' has a value smaller than 30.The second statement successfully processed a statement composed of three items.
-
- The third statement successfully processed 3 items out of five.

Complex statements only provide a summarized status of the execution. The detailed information is nevertheless provided on the Item Information Dialog, which one is accessible by double-clicking one of the item information columns of the statement.



Item information for TM\verify#0
Line: 47 (executed 1 times)

☑ Show latest information

EX	Item Name	Value	Status	Time	Stack	Comments
1	FCOUNTER	<30	SUCCESS	2010-04-12 13:24:19.837	TM\verify	
1	PARAM1	>0	SUCCESS	2010-04-12 13:24:19.884	TM\verify	
1	PARAM2	>0	SUCCESS	2010-04-12 13:24:19.931	TM\verify	
1	PARAM3	<0	FAILED	2010-04-12 13:24:20.337	TM\verify	Actual value: 0.596108974354
1	PARAM4	>0	SUCCESS	2010-04-12 13:24:20.040	TM\verify	

Figure 6: Item information dialog

The dialog splits the detailed information for each item of the statement into several columns:

- **"EX" column**: indicates how many times the item has been executed or evaluated. The figure represents the actual number of iterations. The global amount of iterations for a given line appears at the top of the dialog. In the example, the line has been "executed 1 time(s)".
- **Item name**: is the name of the item, which can be a TM point, a telecommand, etc.
- **Item value**: is the current value of the item, which can be a TM point value, or a telecommand execution stage, amongst other things.
- **Item status**: is the status of the item, which can be SUCCESS, IN PROGRESS, FAILED, or other values for command execution stages.
- **Time**: is the timestamp of the latest change in the row.
- **Stack**: is the unique identifier of the procedure line.
- **Comments:** when applicable, contains extra information about the notification.

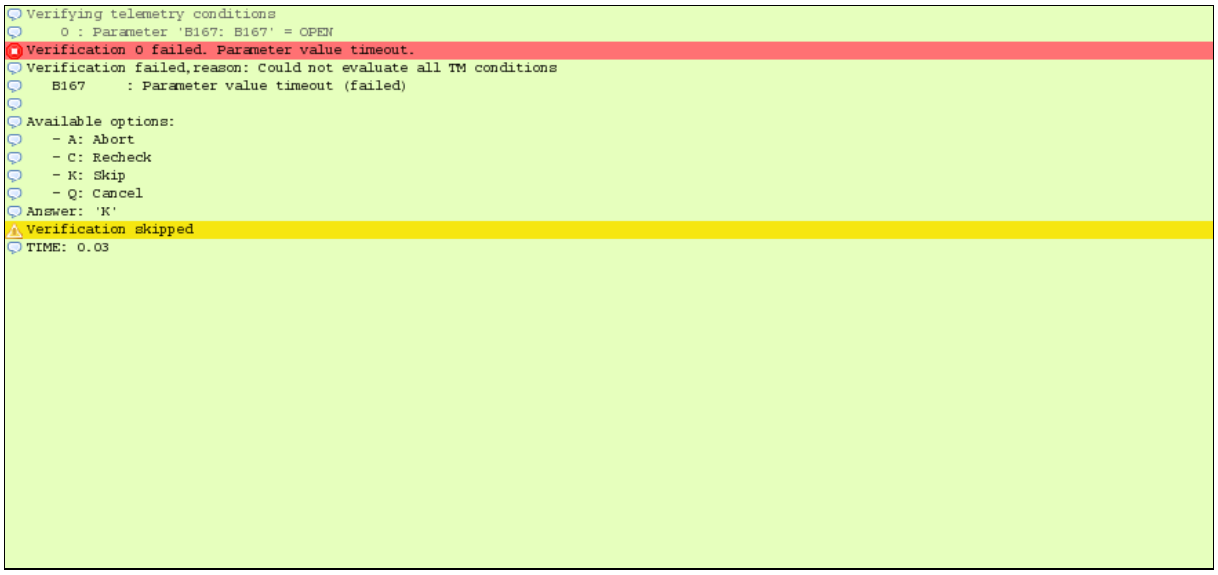
As previously mentioned, the item information dialog shows the status of the execution or iterations of several items. By default, only the information regarding the latest iteration is displayed. All the details can be obtained by un-checking the "Show latest information" check-box (above the notifications table).

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

3.2.2 The Text presentation

The text presentation only displays progress messages regarding the procedure execution, without the source code. All messages displayed by the procedure are displayed in the text area. The background colour of the message changes, depending on its type (prompt, warning or error message).

The foreground colour indicates the scope of the message (according to the preferences selected by the user - see Scope styles on section 4.2.2). The different scopes are: the SPELL procedure itself, the SPELL core and drivers, the step instruction, etc. In addition, some messages are prefixed with a icon in order to easily identify them.



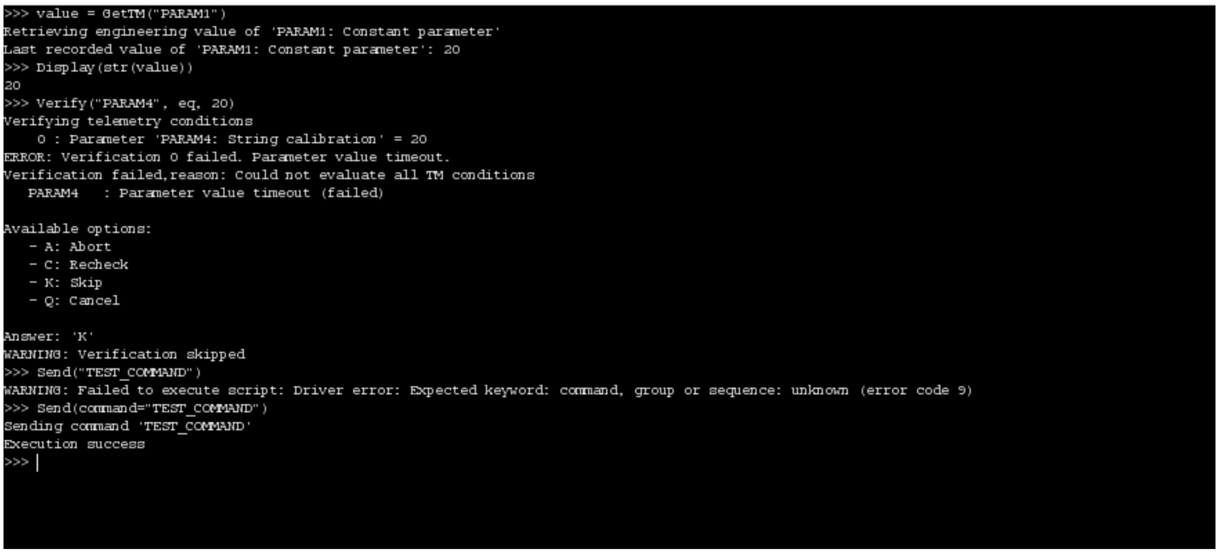
```
Verifying telemetry conditions
    0 : Parameter 'B167: B167' = OPEN
Verification 0 failed. Parameter value timeout.
Verification failed,reason: Could not evaluate all TM conditions
   B167     : Parameter value timeout (failed)

Available options:
   - A: Abort
   - C: Recheck
   - K: Skip
   - Q: Cancel
Answer: 'K'
Verification skipped
TIME: 0.03
```

Figure 7: the Text presentation

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

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. The prompt is represented by the symbol ">>>". When a valid statemnt is entered, the executor processes it and the resulting output is right it.



```
>>> 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

Any SPELL code, including functions calls, can be executed, put apart the goto statement which is not supported at shell level. Multi-line statements such as if clauses, for loops, function definitions and so on are accepted too. The shell is able to detect multi-line statements, and will help the user with automatic indentation. The prompt symbol will then change into "…". To finish a multi-line statement, a blank line shall be entered.

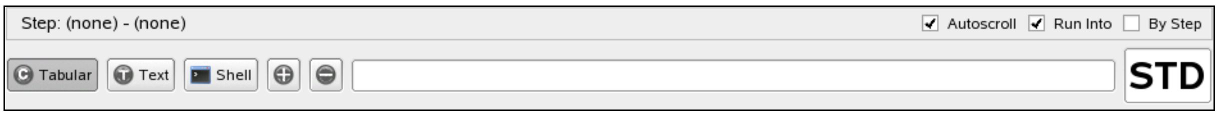
Command history is browsable via the CTRL+Up and CTRL+Down key combination. The user can thus recover previously written commands.

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

Note that the statements executed from the shell interact with the procedure instance. For example, if the procedure execution is stopped within a function call, the code executed from the shell accesses (to get and even modify) the function local variables. Python scope rules apply as if the manual statements were placed inside that procedure function.

3.2.4 Presentations control area

The presentations control area is the fixed frame constantly visible on top of the presentation area itself. It provides the presentation control buttons and displays the current step of the displayed procedure.



Step: (none) - (none)

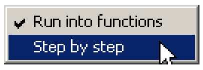
Figure 9: Presentations control area

The step label consists in the current step title and its description, if any. The three check boxes next to it allow controlling the behavior of the presentations as well as the procedure execution flow:

- **Auto-scroll**: enables or disables the auto-scroll feature in all the presentations of the selected procedure. The actual effect of the option depends on the presentation: in the Tabular one, the auto-scroll option makes the executed line always appear in the middle of the area. If the option is disabled, the source code stops scrolling. In the Text presentation, the auto-scroll option makes the text area always scroll down to the end of the area on new line.

- **Run Into**: configures the GUI to follow the execution flow by *stepping into* all functions and sub-procedures. If the run-into option is disabled, the system will *step over* function calls, so that the code inside these functions is executed as a black box. The run-into feature has no effect on the Text presentation.

- **By step**: will make the procedure execution to automatically pause every time a **Step** instruction is executed in the code.

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

Refer to section 5 for information about the procedure execution status and procedure commands.

The presentation control area buttons with the symbols + and – are for zooming. They change the font size of the source code, messages and the step label. The current context is displayed in the large text box next to the zoom buttons. The last text area is used to display the spacecraft the GUI is currently connected to.

3.2.5 The Procedure Control Panel

The procedure control panel is a fixed area visible on all the presentations, allowing the user to control the procedure execution.

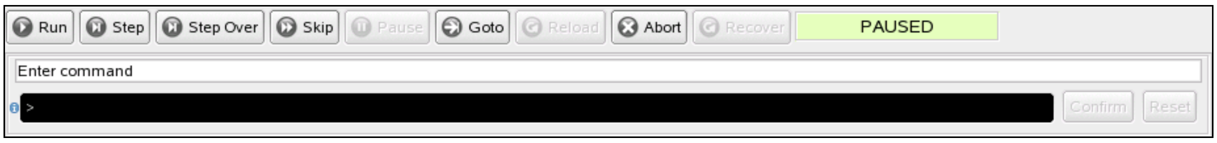


Figure 10: Procedure Control Area

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

The text input area at the bottom can be used to enter the very same commands to control the procedure via the keyboard instead of using the mouse. The information icon on its left provides help for typing the commands.

3.2.6 Prompt Inputs

The prompt area appears below the procedure control panel. This area is hidden when no user input is required and is replaced by the black text box where procedure commands can be typed (e.g. "run", "pause", etc.). Once a prompt is requested, this area changes in order to the prompt the user for input.

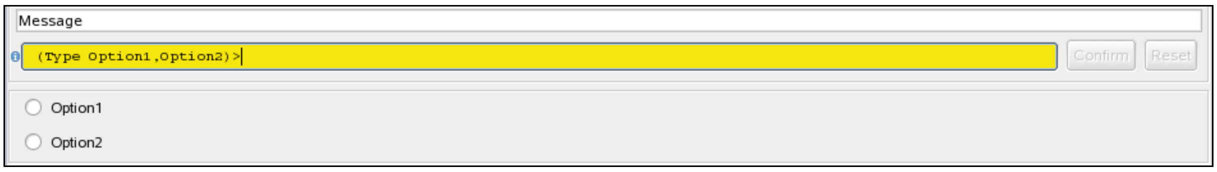


Figure 11: Prompt

Note that the prompt message appears above the text field, and that the text field background turns yellow. An input hint is displayed in the text field when applicable.

For prompts of LIST type, the available answers are displayed below the text field as a set of radio buttons. The answer can be either typed or selected via the radio buttons.

The answer must be validated by clicking the "Confirm" button. The response is then submitted to the procedure and the execution goes on.

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 *aborts the procedure execution.*

GUI prompts have a programmable delay. If set, after a certain amount of time, the prompt area starts blinking with yellow background and a warning sound is optionally played. The delay time and the wave file to be played can be set up on the preference pages.

3.2.7 Other features

Procedure views can be detached from the GUI main window in order to have two windows (or more) of the procedure view 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 on Figure 12.

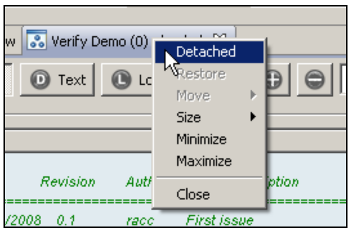


Figure 12: Detaching a procedure view

Once detached, the view appears in a separate window. Note that the procedure can be controlled from that standalone view.

To reattach the view to the main GUI window, simply deselect “Detached” in the same pop-up menu of the view's title tab. The standalone view then goes back to its original place.

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

Minimization is not allowed for procedure views but for following view areas:

- 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 on Figure 13.

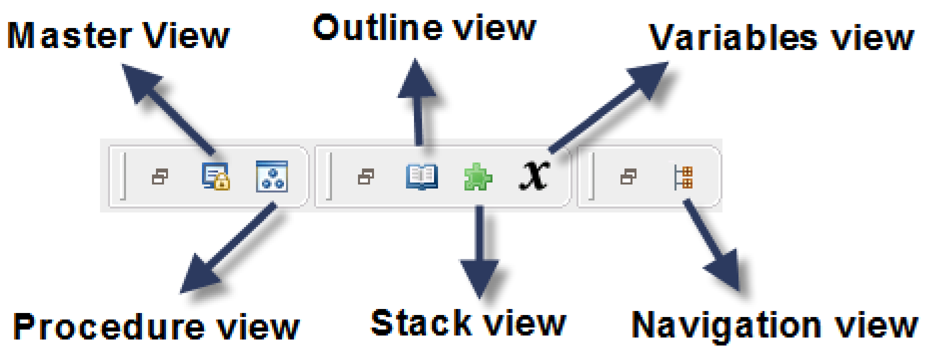
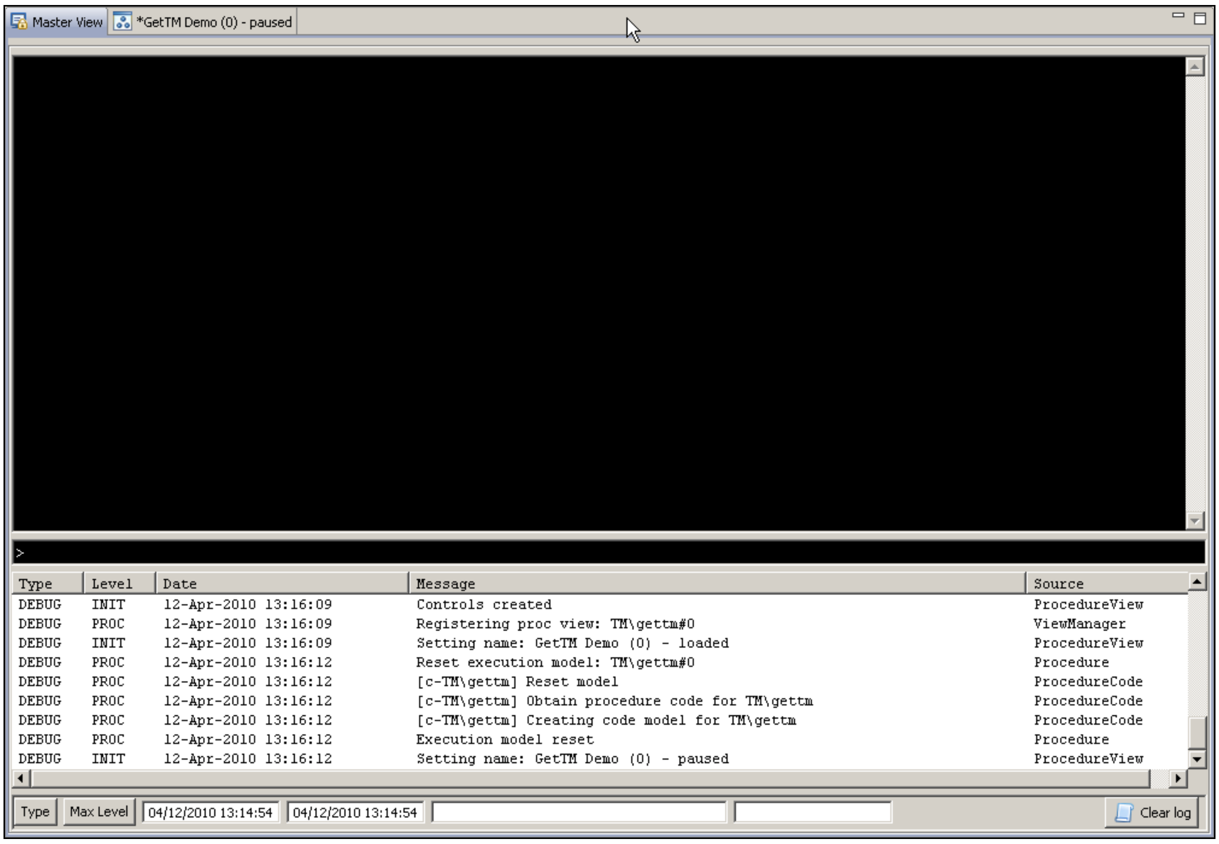


Figure 13: Restore view area icons

3.3 The master view

The master view is permanentlyopen (cannot be closed) and contains two different areas:

- The master console (top part)
- The application log viewer (bottom part)



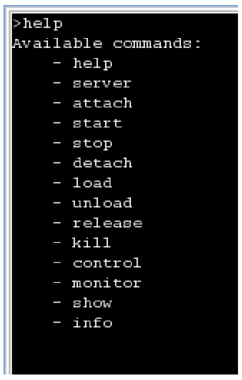
```
Type   Level  Date                  Message                                          Source
DEBUG  INIT   12-Apr-2010 13:16:09  Controls created                                 ProcedureView
DEBUG  PROC   12-Apr-2010 13:16:09  Registering proc view: TM\gettm#0                ViewManager
DEBUG  INIT   12-Apr-2010 13:16:09  Setting name: GetTM Demo (0) - loaded            ProcedureView
DEBUG  PROC   12-Apr-2010 13:16:12  Reset execution model: TM\gettm#0                Procedure
DEBUG  PROC   12-Apr-2010 13:16:12  [c-TM\gettm] Reset model                         ProcedureCode
DEBUG  PROC   12-Apr-2010 13:16:12  [c-TM\gettm] Obtain procedure code for TM\gettm  ProcedureCode
DEBUG  PROC   12-Apr-2010 13:16:12  [c-TM\gettm] Creating code model for TM\gettm    ProcedureCode
DEBUG  PROC   12-Apr-2010 13:16:12  Execution model reset                            Procedure
DEBUG  INIT   12-Apr-2010 13:16:12  Setting name: GetTM Demo (0) - paused            ProcedureView
```

Figure 14: Master view

3.3.1 Master console

All the operations that can be performed using the toolbar buttons, menu commands, etc. can be carried out from the master console. The master console provides supports the following 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
```

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
`help`	`<command>`	Show help for a given command
`Help`		Show the list of available commands

3.3.2 Log viewer

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

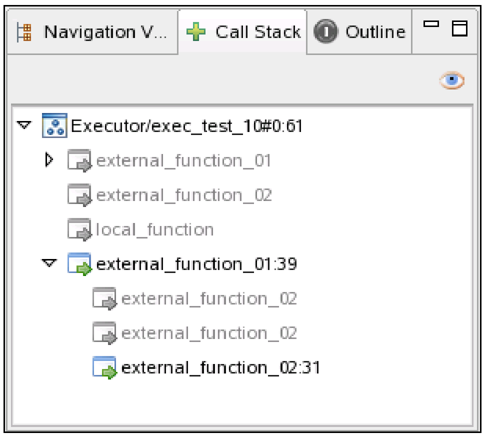
- Message type can be `INFO`, `WARNING`, `ERROR` and `DEBUG`.
- Message levels can be `INIT` (initialization), `PROC` (processing), `GUI` and `COMM` (communications)

3.4 The call stack view

The call stack area traces information about the subroutines executed inside a procedure at runtime.

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

The SPELL GUI provides a view in its workbench which helps to keep track of the execution procedure through the stack. Subroutines are displayed as a tree, each node representing one of them.

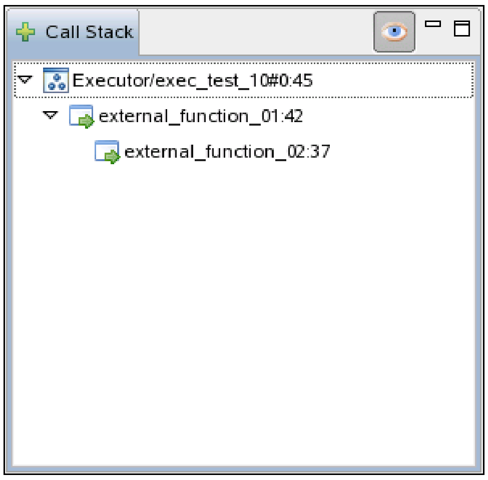


```
Executor/exec_test_10#0:61
  external_function_01
  external_function_02
  local_function
  external_function_01:39
    external_function_02
    external_function_02
    external_function_02:31
```

Figure 15: Call stack view

The root node of the call stack tree is the procedure identifier, indicating the main routine and the current line number. Each child represents a subroutine, which is displayed in black when active or gray when inactive. Next to the active one also appears the line currently being executed. If the caller of the subroutine already is a subroutine, the called element appears as child element of the calling subroutine.

Inactive nodes can be hidden in the view by clicking on the eye button located at the top right. These nodes will remain hidden as long as the button remains toggled.

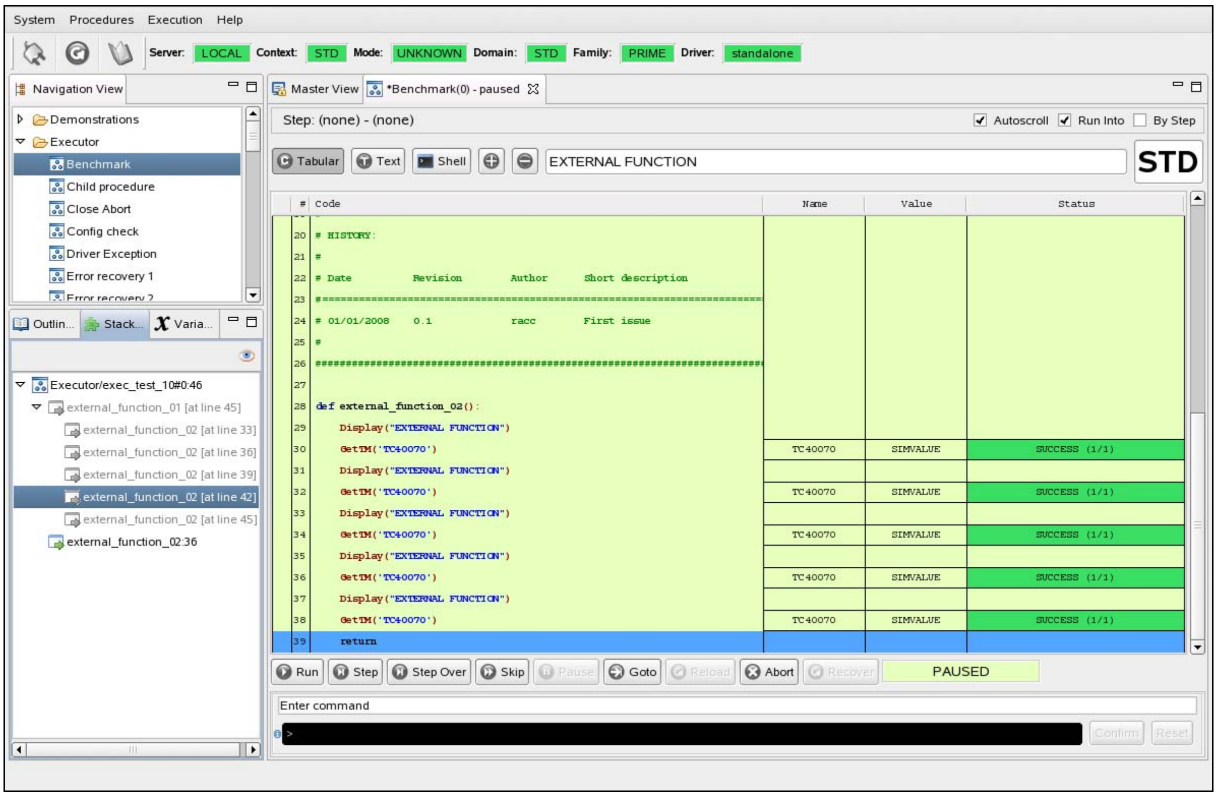


```
Executor/exec_test_10#0:45
  external_function_01:42
    external_function_02:37
```

Figure 16: Hiding inactive nodes

The nodes of the call stack tree also allow to navigate through the different levels of the running procedure. The Procedure Code presentation is instantly refreshed with the code of the selected node . Navigate through the execution history of the procedure also shows the received data and the performed operations at the moment it has actually been executed. Selection of the call stack node is done by double-clicking on the item of the call stack view.

The navigation feature is only available when the procedure is in PAUSED, WAITING or ERROR states. If the procedure is switched to STEP or RUN mode, the code view is automatically reset to the currently executed code.



Server: LOCAL Context: STD Mode: UNKNOWN Domain: STD Family: PRIME Driver: standalone

Step: (none) - (none)   [✓] Autoscroll [✓] Run Into [ ] By Step

EXTERNAL FUNCTION

PAUSED

```
Executor/exec_test_10#0:46
    external_function_01 [at line 45]
        external_function_02 [at line 33]
        external_function_02 [at line 36]
        external_function_02 [at line 39]
        external_function_02 [at line 42]
        external_function_02 [at line 45]
    external_function_02:36
```

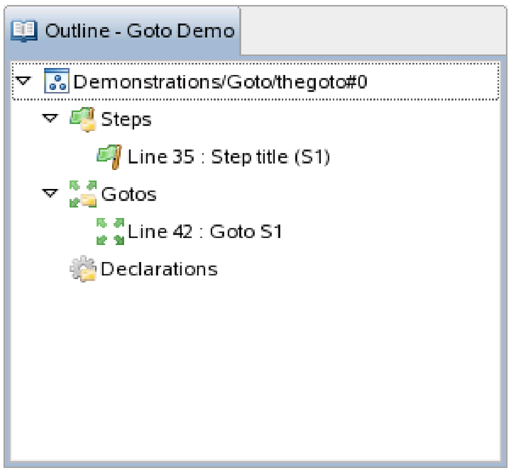
#	Code	Name	Value	Status
20	# HISTORY:			
21	#			
22	# Date Revision Author Short description			
23	#===			
24	# 01/01/2008 0.1 racc First issue			
25	#			
26	##			
27				
28	def external_function_02():			
29	Display("EXTERNAL FUNCTION")			
30	GetTM('TC40070')	TC40070	SIMVALUE	SUCCESS (1/1)
31	Display("EXTERNAL FUNCTION")			
32	GetTM('TC40070')	TC40070	SIMVALUE	SUCCESS (1/1)
33	Display("EXTERNAL FUNCTION")			
34	GetTM('TC40070')	TC40070	SIMVALUE	SUCCESS (1/1)
35	Display("EXTERNAL FUNCTION")			
36	GetTM('TC40070')	TC40070	SIMVALUE	SUCCESS (1/1)
37	Display("EXTERNAL FUNCTION")			
38	GetTM('TC40070')	TC40070	SIMVALUE	SUCCESS (1/1)
39	return			

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

3.5 The Outline view

The outline view of the SPEL-GUI describes the code displayed in the currently selected Procedure View. This contents includes 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 they are declared.



```
Outline - Goto Demo
▽ Demonstrations/Goto/thegoto#0
    ▽ Steps
        Line 35 : Step title (S1)
    ▽ Gotos
        Line 42 : Goto S1
    Declarations
```

Figure 18. Outline view

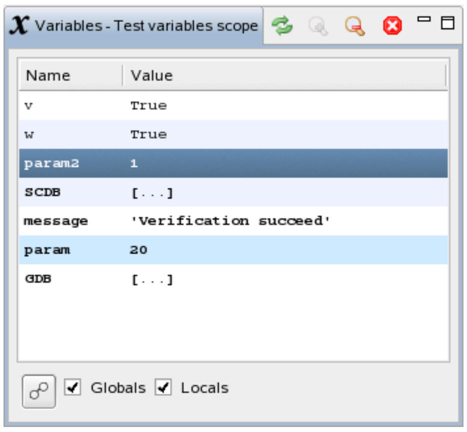
3.5.1 Interacting with the outline view

Double clicking any element of the outline view will automatically scroll the Code Presentation to the corresponding element of the source code.

This feature is only enabled when the procedure execution is paused. Double clicking on the outline view has no effect while in running mode.

3.6 The Variables view

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



Variables - Test variables scope

Name	Value
v	True
w	True
param2	1
SCDB	[...]
message	'Verification succeed'
param	20
GDB	[...]

☑ Globals ☑ Locals

Figure 19. Variables view

The outline view presents two types of variables: the local variables, which have been declared inside the current execution frame, and global variables, which have been declared in the main scope but can be accessed from anywhere. Local variables are displayed in this view using normal font, whilst global variables are in bold.

Users can choose whether to show or hide local and global variables by changing the selection of the buttons labeled "Globals" and "Locals".

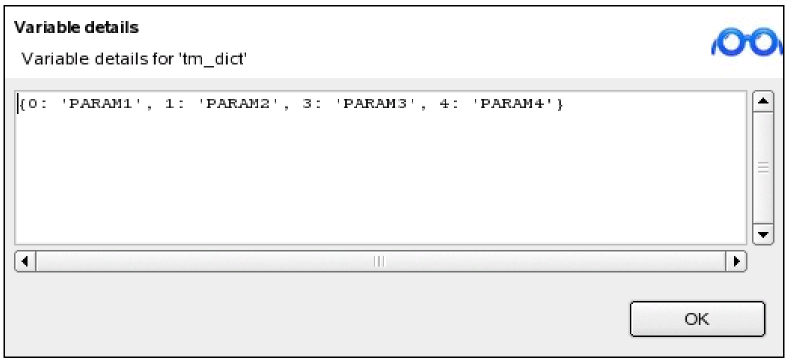
Note that for the performance's sake, variable changes are not automatically reflected at procedure execution time. In order to update them, an explicit request shall be issued, except for the currently watched variables as explained in the following section.

3.6.1 Checking variable values

During the procedure execution, users can update the variable values of the Variables view by clicking the refresh button.

Figure 20. Refresh button icon

If the length of the variable's value is too long to fit in the available area, the symbol "[…]" is then displayed and users may see the details by double clicking on the variable name. A new dialog with the contents of the variable will appear.



Variable details

Variable details for 'tm_dict'

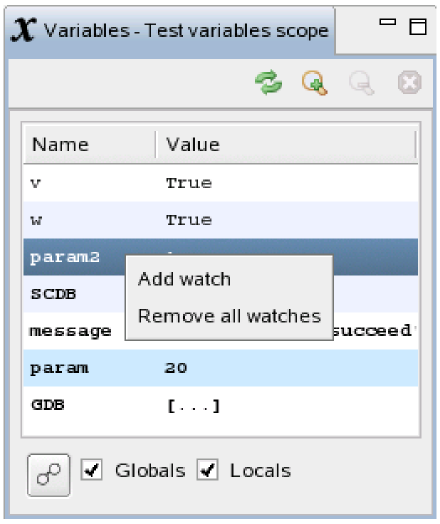
```
{0: 'PARAM1', 1: 'PARAM2', 3: 'PARAM3', 4: 'PARAM4'}
```

Figure 21. Variables detail dialog

3.6.2 Watch of variables

It is also possible to select the specific variables or set of variables to follow-up. This is named "variable watch" and can be created via the variables view. Any watched variable is monitored from the GUI, and value changes are automatically detected and highlighted.

A variable watch is created by selecting a variable on the view table and clicking the "Watch variable" button. Alternatively the "Watch variable" option of the table contextual menu can be used.



Name	Value
v	True
w	True
param2	
SCDB	
message	succeed'
param	20
GDB	[...]

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

Watch variables can be discarded 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

To disable all watches at once, use 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 tune up the execution to make the procedure successfully evaluate an "if" condition and execute the statements within it for instance.

There are several ways of changing the value of a variable. This can be done directly from the variables view, or

If the procedure status is paused, users can modify a value by clicking on the value field. A text editor then appears at the value's location. If the value is modified and confirmed by pressing ENTER, the variable's value is then updated. Pressing the ESC key will leave the variable's value unchanged.

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

- By typing the value directly. Valid values are for instance `1`, `‘String’`, `[1,2,3,4]`, `{‘key’ : ‘value’}`. It is similar as setting Python variables with an explicit value.

- By calling functions. For instance `range(0,10)` would create a list of 10 elements starting from 0. When set this way, the value column in the variables view displays the result of the evaluated function. In the given example, the Value column would contain the list `[0,1,2,3,4,5,6,7,8,9]`

3.7 The status bar

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

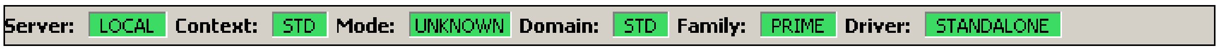


Figure 26: Status bar

It shows information about:

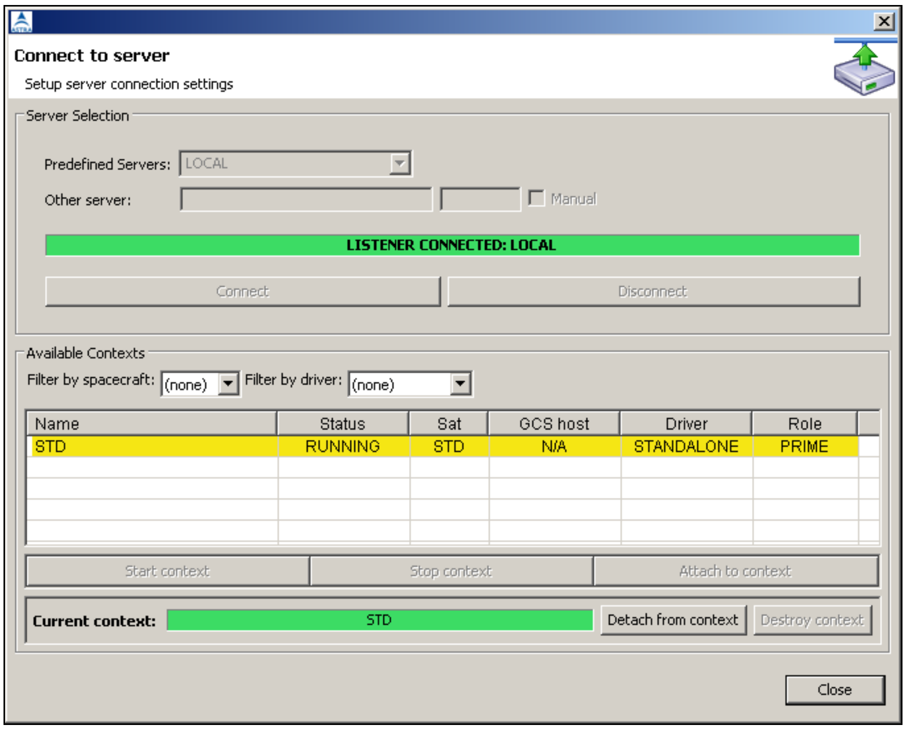
- The Server: the current SPELL server being used
- The Context: the current context being used
- The GUI mode: commanding or monitoring
- The Domain: the spacecraft name
- The Family: when applicable, depending on the GCS, PRIME or BACKUP
- The Driver: the name of the SPELL driver being used.

Fields that appear with a green background are nominal. A yellow color indicates a pending status and the red color a failure.

3.8 The connection dialog

The connection dialog is used to manage the GUI connections with the SPELL server. It is divided in two sections: one for the Listener and the second for the Contexts.

Connection to a listener can be done either by selecting it into a predefined list, or manually by specifying the address and port of the target listener and clicking the “Connect” button to establish it. Upon successful connection, the information area just above the buttons becomes green and displays the name of the listener being used.


Connect to server
Setup server connection settings

Server Selection

Predefined Servers: LOCAL
Other server: ☐ Manual

LISTENER CONNECTED: LOCAL

Available Contexts
Filter by spacecraft: (none) Filter by driver: (none)

Name	Status	Sat	GCS host	Driver	Role
STD	RUNNING	STD	N/A	STANDALONE	PRIME

Current context: STD

Figure 27: Connection dialog

Once connected to the listener, the list of available SPELL contexts appears. Each context is presented with the following information:

- The context name
- The context status (AVAILABLE / RUNNING)
- The corresponding spacecraft
- The GCS host name
- The SPELL driver
- The GCS host role or family (PRIME/BACKUP)

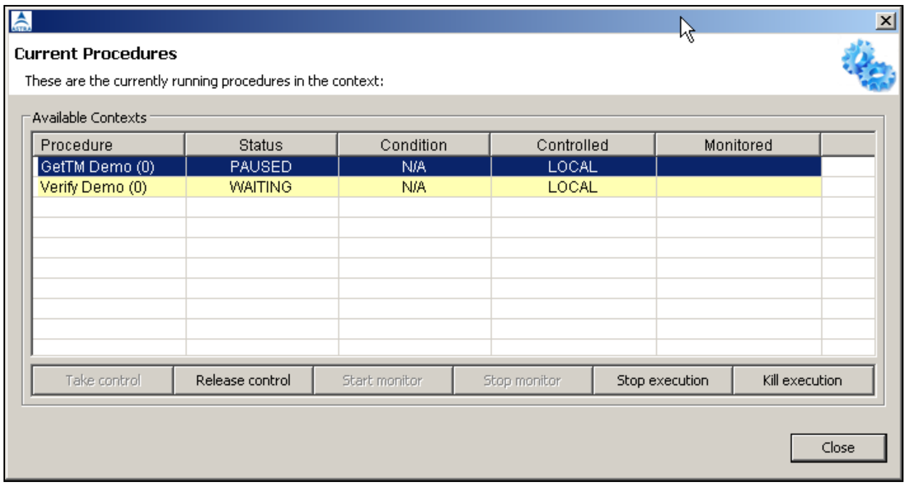
The yellow background indicates that the context is already running. The GUI can attach or detach to these contexts by selecting them and clicking the buttons “Attach to context” or “Detach from context”.

The GUI can only attach to a started context. If it is not running, this can be performed by selecting it and clicking the “Start context” button. On the other hand, it is only possible to stop a context if nobody is attached to it by using the “Stop context” button. The “Current context” area turns green as soon as the GUI is attached to a valid context.

The "Destroy context" button can be used to kill a context process in case of hang-up. Note that this is not a graceful stop and may leave the GCS in an inconsistent state.

3.9 The executors dialog

The executors dialog provides information on all the procedure instances of the current SPELL Context. These instances do not need to belong to the current GUI. They may be controlled by any other client working within the same context.


Current Procedures

These are the currently running procedures in the context:

Available Contexts

Procedure	Status	Condition	Controlled	Monitored
GetTM Demo (0)	PAUSED	N/A	LOCAL	
Verify Demo (0)	WAITING	N/A	LOCAL	

Figure 28: Executors dialog

The list of this dialog shows the following information for each of the procedure instances (named executor):

- The executor name (with instance number)
- The status of the execution
- The client in control of the execution, if any
- The list of clients monitoring the execution, if any.

From this dialog the following actions can be carried out:

- To take control of an ongoing procedure execution: provided that there is no other client already controlling it (see “Controlled” column)

- To release the control of a procedure. This can only be done if the procedure is authorized to run in background mode and if the GUI is in control.

- To start monitoring a procedure: the GUI will show the execution information but it will not be possible to issue any command to the procedure or to provide any user input.

- To stop monitoring a procedure, if the GUI is monitoring it.

- To stop execution: only available if the GUI is in control.

- To kill execution: forces the procedure closure in any case, provided that the GUI is the controlled client. This is not a graceful stop and might leave the GCS in a inconsistent state.

It is possible to select several executors at the same time. The buttons are then enabled or grayed out according to the overall characteristics of the executors. For example, the release control button would be enabled *only* if all selected executors are in control of the GUI.

4 Startup and configuration

4.1 Startup

The application needs to be started using launcher scripts. These scripts can be found in the `<SPELL_HOME>/bin` directory. This directory contains several GUI launchers that depending on the platform where the application is being run (Windows or Linux):

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

The SPEL-GUI application then runs with the default configuration, 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:

- The GUI behavior at startup (auto-connection features)
- The GUI appearance (colors)
- The predefined list of listeners (available SPELL servers) with the specific settings to connect to them.

These properties can be changed through the preferences management system. The SPEL-GUI configuration and preferences management are explained in the following sections.

4.2.1 Configuration file

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

```xml
<?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 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`". A set of "`property`" tags appear directly under this tag.

These properties are global options for the GUI application:

- **`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 auto-connection feature
- **`InitialServer`**: if auto-connection is used, this identifies the SPELL server that the GUI will try to connect to automatically
- **`InitialContext`**: if auto-connection is set, this property identifies the SPELL context to which the GUI will attempt to attach to after connecting to the SPELL server. This option may be disabled by setting the value to "NONE". Once the SPELL server connected, the GUI will then 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

The `LastServerConnected`, `LastHostConnected`, `LastHostConnected`, `LastConnectionManual` properties are only set at preferences export. The preferences exporting and importing mechanism is presented in section 4.2.3.

After the set of application property tags, the XML file contains three other main sections: "`presentations`" , "`appearance`" and "`servers`". The first identifies the available procedure presentation plugins, the second contains the GUI color/font configurations, and the last specifies the list of predefined 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 plug-in in the SPELL GUI plug-ins directory, and (2) add a presentation tag in the configuration file.

The presentation tag specifies the presentation name as defined in the plug-in. Only one of the presentations in this section can have the *default* property set to "yes". This presentation will be the one selected 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

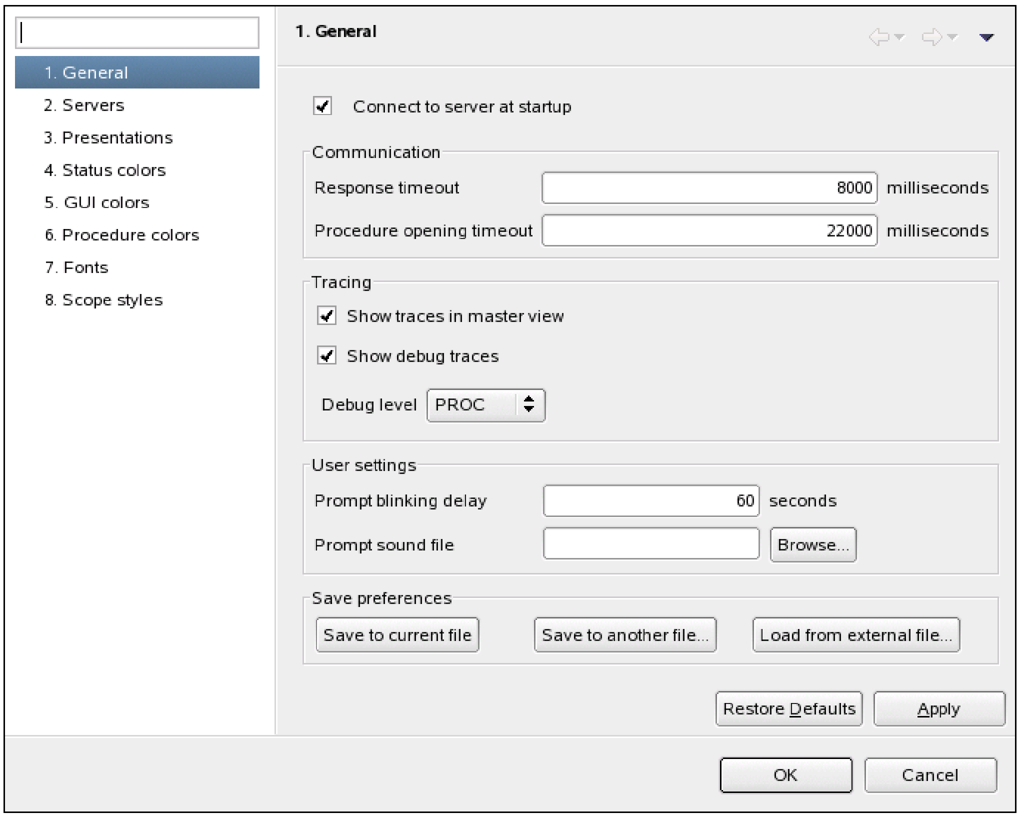
The servers tag contains a set of “`server`” items. Each server is composed of a name, a host name, a role and a port number. The name field may be used in the **`InitialServer`** property for auto-connection purposes.

The role value can be either `COMMANDING` or `MONITORING`. The former makes the GUI that is connecting to this server be able to open and monitor procedures. The latter restricts the GUI to monitor procedures only.

Optionally, a user name and a password may be given, if a SSH tunnel is required to access the server. If the user name is given without no password is given, the user will be prompted for one at runtime.

4.2.2 Preferences management

The properties presented in the previous section can be changed via the preferences management system. This is accessed through the System menu at the top of the main application window.


1. General
2. Servers
3. Presentations
4. Status colors
5. GUI colors
6. Procedure colors
7. Fonts
8. Scope styles

1. General

[✓] Connect to server at startup

Communication
Response timeout 8000 milliseconds
Procedure opening timeout 22000 milliseconds

Tracing
[✓] Show traces in master view
[✓] Show debug traces
Debug level PROC

User settings
Prompt blinking delay 60 seconds
Prompt sound file

Figure 29. Preferences dialog

The dialog presents eight pages, that can be selected by clicking on page name on the left side of the dialog. Each page corresponds to one configuration file section, and allows changing its values in an easy way. The two buttons at the bottom of each page can be used to store the new values or restoring the defaults, respectively labeled Apply and Restore Defaults**.**

The OK and Cancel button at the very bottom of the dialog either store the values of the dialog or not, before closing the dialog

The available pages of the 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**: To let users change the status colors.
- **GUI colors**: To let users change the GUI colors.
- **Procedure colors**: To let users change the Procedure status colors.
- **Fonts**: To change the application fonts.
- **Scope styles**: To change the scope message styles.

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 is performed through the preferences dialog.

Three different buttons are available on the General preferences page:

- **Save to current file**: The file used for loading preferences will be overwritten with the current preference values.
- **Save to external file**: The current preference values are stored in a file specified in the file dialog prompt that pops up. If the selected file already exists, the user is prompted for confirmation to overwrite that file.
- **Load from external file**: Loads a file specified in the file dialog prompt. The current preferences are overwritten with the values stored in the file.

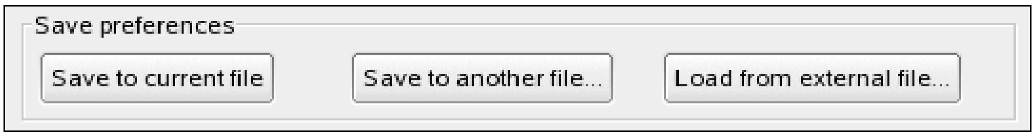


Figure 30. Import/Export preferences section

4.3 Initial steps

In order to prepare a SPELL execution session:

a) Open the connection dialog
b) Connect to a listener
c) Select and start the desired context, if not already running
d) Select and attach to the context
e) Close the connection dialog
f) Load procedures

5 Procedure Execution

5.1 Procedure execution status

Procedures run in what is call 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 presents the list of possible procedure status with their description. The "Default color" column shows the color used in the GUI code and textual pages of a procedure view to indicate the current procedure status.

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

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 authorize 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 directly sent to the GCS, provided that this type of communication is possible (SPELL event service must be available in the driver).

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

A background procedure becomes a foreground procedure when a SPELL clients takes control of the execution.

5.3 Procedure execution control

Procedure execution is controlled using the control area buttons.

The following actions or commands are available:

- ***Run***: executes the procedure without pausing.
- ***Step***: also known as step-into. Executes one statement only, pausing at the first one of the function in the case of a function call. If the executed statement calls a function provided by a sub-procedure, the procedure code will be substituted by the sub-procedure code. When returning from the sub-procedure's function, the caller's procedure code will be displayed again. This applies to the Code Page only.
- ***Step Over***: same as step, but pauses at the next statement of the current function. If the executing the last statement of the current function, the caller's source code is displayed.
- ***Skip***: skips the statement and pauses at the next one.
- ***Pause***: pauses the procedure execution. Applies in running mode.
- ***Goto***: goes to a given Step or line of the procedure.
- ***Reload***: reloads the procedure.
- ***Abort***: aborts the procedure execution.
- ***Recover***: recovers the procedure from an abnormal ending.

Note that depending on the procedure status, some buttons may be disabled since not applicable. The following table shows the command availability with regards to 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 previously explained, the control area is used to prompt for user input.

Whenever the `Prompt()` function is executed in a procedure, the prompt appears in this area. The figures below show examples of how the input area may look like when a selection or text prompt is executed.

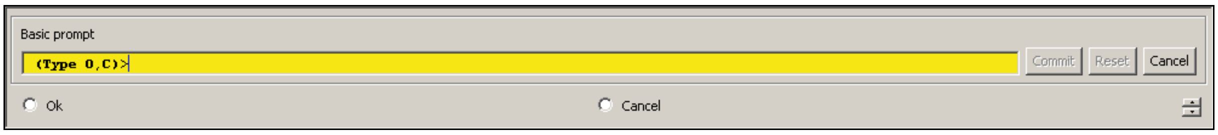


Figure 31: Input area with selection prompt

Selection prompts consist on a list of predefined choices at procedure's level for the user. The options are displayed using radio buttons under the text field of the input area (a scroll bar appears when necessary). The answer can be given via the keyboard as well. The text field provides a hint regarding the expected answers. In the example below, the prompt provides "O,C" as a hint. Only those letters will be accepted.

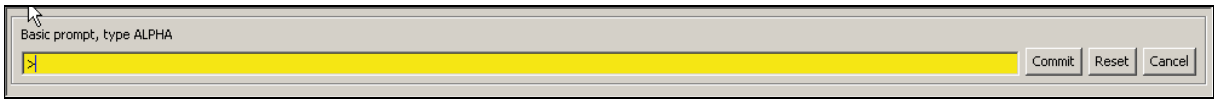


Figure 32: Input area with text prompt

Text prompts accept any alphanumerical input. Only the text field is displayed. The procedure can also specify restrict the valid answers to a given list. Numerical prompts only accept decimal numbers.

Three buttons are available in the input area: “Commit”,”Reset” and “Abort”. The “Commit” button submits the user's input to the procedure. The “Reset” button deselects all the elements and clears any typed text. The “Abort” button cancels the prompt operation. Note that to *cancel 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, by clicking the Goto control button in the procedure view. A Goto dialog then appears:

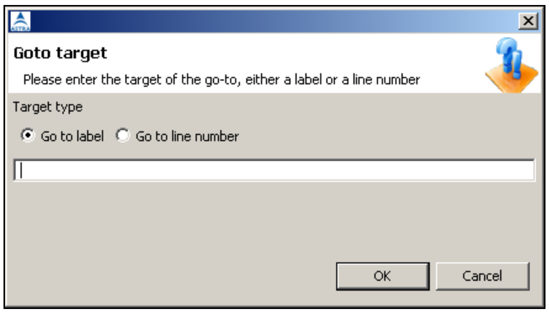

Goto target
Please enter the target of the go-to, either a label or a line number
Target type
Go to label
Go to line number
OK
Cancel

Figure 33: Goto dialog

The target line can be identified either by a label or by a line number. Valid labels correspond to the **Step** statements defined in the procedure source code.

Due to Python scope rules, it is nevertheless not possible to perform a go-to jump to an arbitrary line. If the current line is located within the scope of a function, the go-to target must remain within the scope of that function. The same rule 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 functions, from main code into a function, or from one function to the main code. The same applies for cross-procedures jumps..

5.6 Monitoring and control

When loading a procedure, the GUI that summoned it implicitly has its control.It is then able to send commands to it and control the execution workflow.

Other GUIs may look at the procedure execution, but cannot send commands to it. Those GUIs are *monitoring* the procedure.

In order to monitor a procedure, the user selects it on the Executors Dialog and clicks the "Start monitor" button. The procedure view will appear in the GUI, but the control area will be disabled. Closing the view by selecting "Stop monitor" in the Executors Dialog ends up the monitoring session.

5.6.1 User handover

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

5.7 Scheduling procedures

It is also possible to postpone procedures execution, by selecting it from the navigation view and clicking the *Procedures/Schedule selected procedure* menu. The procedure scheduling configuration dialog then appears:

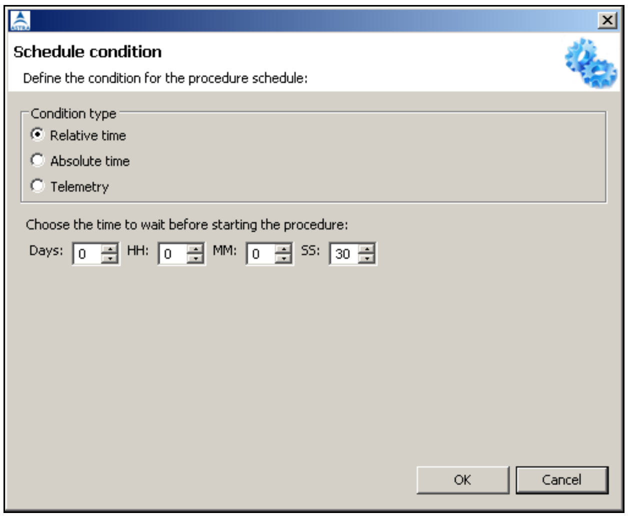
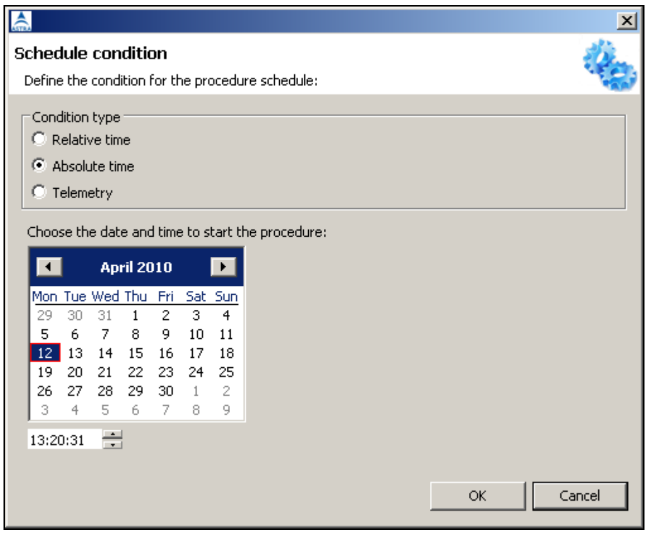

Schedule condition
Define the condition for the procedure schedule:
Condition type
Relative time
Absolute time
Telemetry
Choose the time to wait before starting the procedure:
Days: 0
HH: 0
MM: 0
SS: 30
OK
Cancel

Figure 34: Schedule condition dialog

There are three possibilities for scheduling a procedure: to wait for a given relative time , to wait until a given absolute time , or to wait for a set of telemetry conditions to be fulfilled before starting execution. By default, the relative time option is selected.

The relative time condition is specified giving number of days, hours, etc. to wait before starting the procedure execution. The absolute time condition requires selection of a specific date and time:



Schedule condition

Define the condition for the procedure schedule:

Condition type
( ) Relative time
[✓ Absolute time]
( ) Telemetry

Choose the date and time to start the procedure:

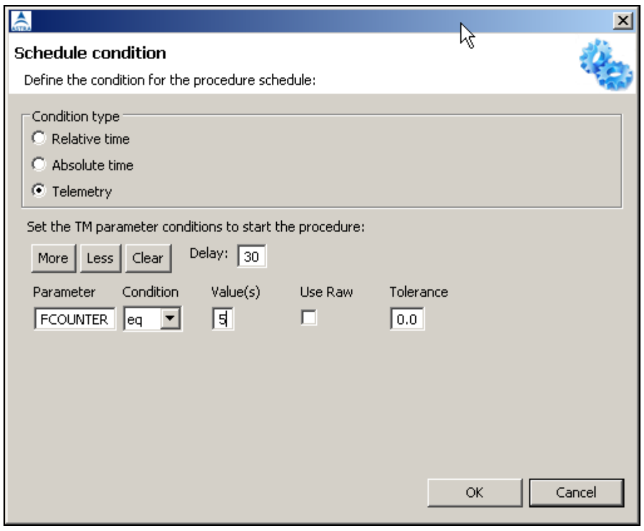
April 2010

Mon	Tue	Wed	Thu	Fri	Sat	Sun
29	30	31	1	2	3	4
5	6	7	8	9	10	11
[✓ 12]	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

13:20:31

Figure 35: Scheduling with absolute time condition

The telemetry condition is set selecting a set of TM points and indicating the conditions that must be fulfilled to start the procedure execution:



Schedule condition

Define the condition for the procedure schedule:

Condition type
( ) Relative time
( ) Absolute time
[✓ Telemetry]

Set the TM parameter conditions to start the procedure:

Delay: 30

Parameter	Condition	Value(s)	Use Raw	Tolerance
FCOUNTER	eq	5	☐	0.0

Figure 36: Telemetry scheduling condition

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

Once the condition set, the procedure view appears and the procedure remains in WAITING state, until the scheduling condition is fulfilled. The procedure then automatically switches to RUNNING mode.

5.8 Procedure files

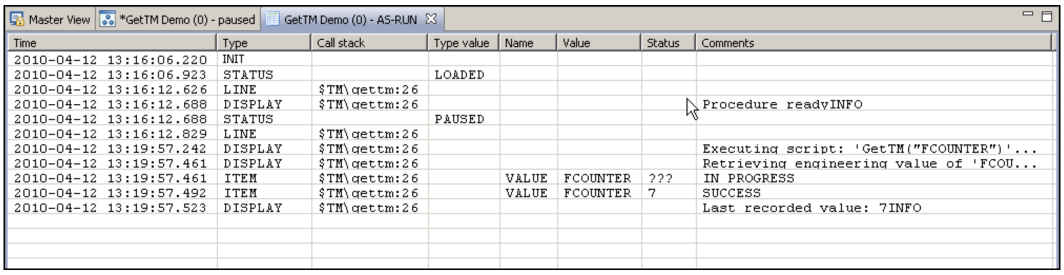
Some files are being generated during the procedure execution, located on the SPELL server side.

5.8.1 As-Run files

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

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

This view is static, meaning that the information is not refreshed while the procedure is executing . The user needs to deselect and select the As Run view again (e.g. the procedure view tab is selected, then the As-Run view tab is selected again).



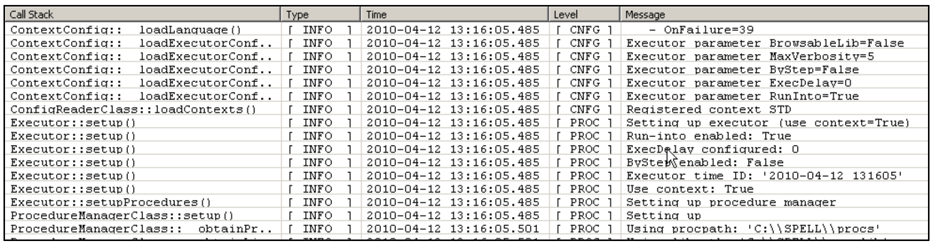
Time	Type	Call stack	Type value	Name	Value	Status	Comments
2010-04-12 13:16:06.220	INIT						
2010-04-12 13:16:06.923	STATUS		LOADED				
2010-04-12 13:16:12.626	LINE	$TM\gettm:26					
2010-04-12 13:16:12.688	DISPLAY	$TM\gettm:26					Procedure readyINFO
2010-04-12 13:16:12.688	STATUS		PAUSED				
2010-04-12 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: 7INFO

Figure 37: As-Run view

5.8.2 Log files

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

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



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

Figure 38: Procedure log file view

This view is static, meaning that it is not refreshed while the procedure goes on running. As for the As Run view, the user needs to deselected and selected the log view 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 works only with the Code Presentation.

The breakpoint mechanism allows setting checkpoints in the source code where the procedure execution automatically pauses

As described in the next sections, there are two different types of breakpoints: the permanent and the temporary breakpoints. The permanent breakpoint remains active when the reached. Once reached and switched to paused state, the temporary breakpoint is discarded.

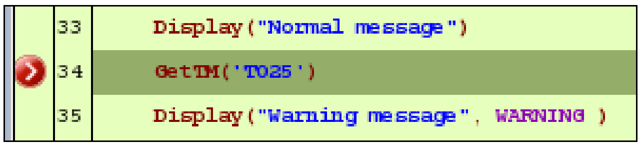
Only permanent breakpoints can be explicitly set by the user.

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 provides different options including three breakpoint operations:

- Add/Remove a breakpoint: to set a breakpoint at the selected line
- Remove all breakpoints: to discard all the breakpoints of the procedure

To add a breakpoint, select the “Add breakpoint at this line” option. A red bullet then appears at the beginning of the line. To remove the breakpoint, select the “Remove breakpoint” option from the same menu.



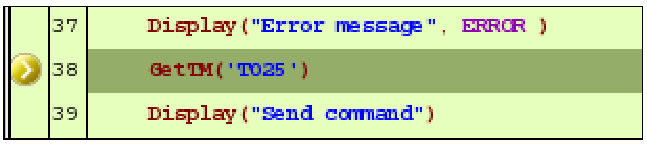
```
33    Display("Normal message")
34    GetTM('T025')
35    Display("Warning message", WARNING )
```

Figure 39: Line with a breakpoint

Note that all the defined breakpoints are automatically discarded when the procedure is reloaded.

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

It is possible to run a procedure until a selected line. To do this, select the “Run until this line” option of the Code view contextual menu. A yellow bullet appears at the beginning of the line, indicating the temporary breakpoint. The procedure then starts running, until the line with temporary breakpoint is reached; it then pauses and the temporary breakpoint is discarded.



```
37    Display("Error message", ERROR )
38    GetTM('T025')
39    Display("Send command")
```

Figure 40: Execution will pause at line 38

6 Other features

6.1 Printing

Several printing options are available:

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

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

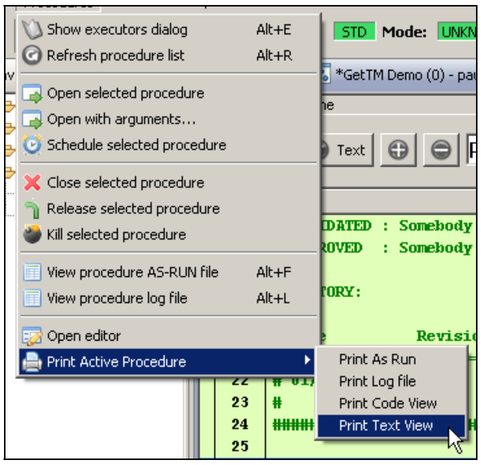
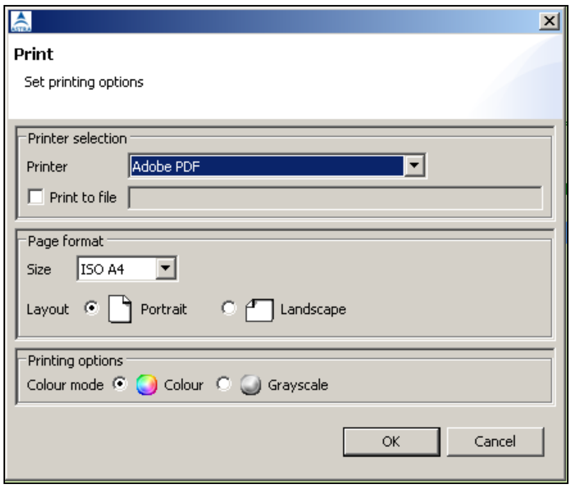


Figure 41: Printing procedure data

The last four operations mentioned can be performed with the corresponding menu commands. When printing any of them, the print dialog pops up:


Print
Set printing options

Printer selection
Printer: Adobe PDF
[ ] Print to file

Page format
Size: ISO A4
Layout: [✓ Portrait] Landscape

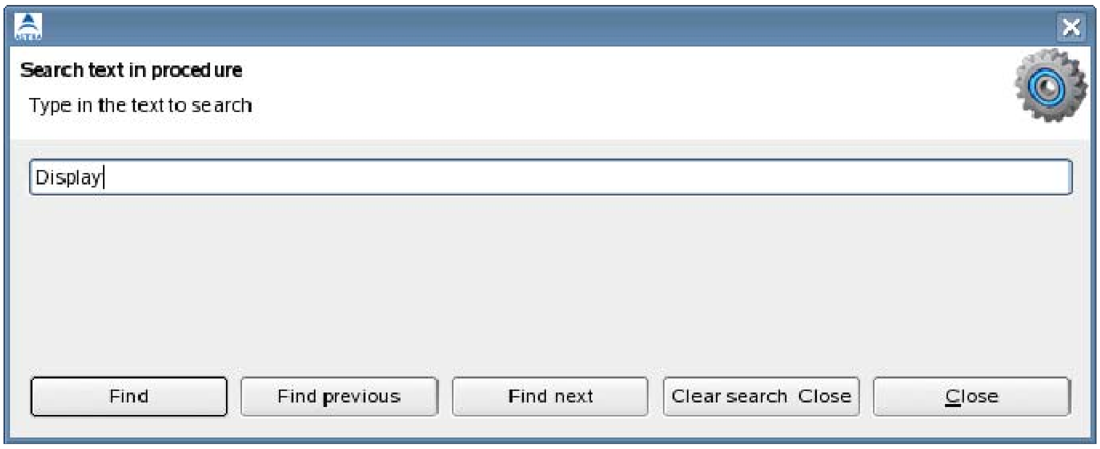
Printing options
Colour mode: [✓ Colour] Grayscale

OK Cancel

Figure 42: Printing dialog

6.2 Code search

When the code view is selected, the option for searching text in the code is available. The contextual menu then provides different options. The Search option pops up a dialog allowing the user provide the text to search.

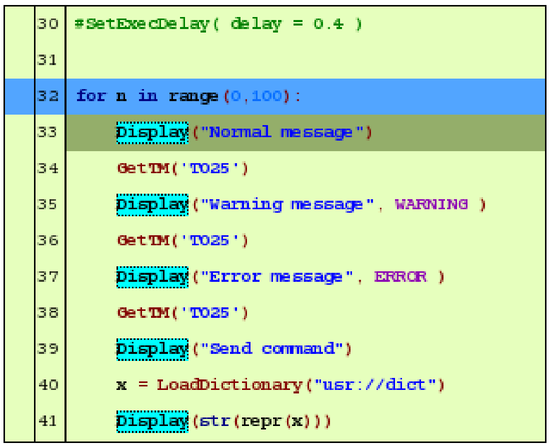


Search text in procedure

Type in the text to search

Display

Figure 43: Text search dialogThe Find button in the Search dialog makes all the text occurrences in the source code become highlighted, and the line with the first hit is automatically selected. The next and previous hits can also searched for by using the corresponding buttons.



```
#SetExecDelay( delay = 0.4 )

for n in range(0,100):
    Display("Normal message")
    GetTM('T025')
    Display("Warning message", WARNING )
    GetTM('T025')
    Display("Error message", ERROR )
    GetTM('T025')
    Display("Send command")
    x = LoadDictionary("usr://dict")
    Display(str(repr(x)))
```

Figure 44: Search results are highlighted

The clear search close option in the dialog is selected, the dialog is closed and the search highlighting disappears. If the user only closes the dialog, the search highlighting remains. It can be removed later on selecting the Clear search close option, or selecting 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 for pasting in an external editor. The copy options is accessible via the code view contextual menu.

7 Getting help

The top menu Help is available in SPEL-GUI, which allows checking the SPELL GUI version in use, the current release information, and gives access to the documentation.

When selecting this menu, four options are available:

- **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 RCP log file.
- **Release information**: When this option is selected, a dialog describing the new features added in the on use SPELL GUI instance pops up.
- **SPELL Language manual**: This option launches an external PDF document viewer displaying the SPELL language document, which describes all the SPELL functions and their behavior.
- **SPELL GUI manual**: This option launches an external PDF document viewer, displaying this document.