SPELL Server Manual

SPELL version 2.4.4

Distribution list

Full Report:
SES-SSO-SOE

For Information Purposes:
Open Source

Reference:
SES-SSO-SOE-SPELL-2015/05

Date issued:
February 2015

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

SES

SPELL version 2.4.4

Table of Contents

1 Introduction

1.1 Purpose of this document

This document is the SPELL server user manual for the SPELL framework. It is intended to be used by software teams in order to install and configure the SPELL Server component (part of the execution environment) of the SPELL framework, on a Linux platform.

1.2 Software Requirements

The following packages and libraries are needed to run the SPEL-Server component:

Package	Version	Remarks
GNU autotools	1.9	
GNU GCC buildchain	4.1.2	
GNU Bash	3.1	
GNU Binutils	2.16	
GNU Make	3.8	
POSIX threads (libpthread)	2.4	
POSIX 1.b realtime extensions (librt)	2.4	
BSD utility functions (libutil)	2.4	
Xerces-C (libxerces-c)	27	
Dynamic linking interface (libdl)	2.4	
C/C++ standard libraries (libstd)	6.0	
Python library and headers (libpython)	2.5,2.6	**Not compatible with 2.7+**
NCurses development libraries (libncurses) (only Executor Cmd Line)	5.5	
Log4CPlus	1.2	log4cplus.sourceforge.net
Subversion(for 'svn' databases only)	<1.7.0	

1.3 Installation basics

A SPELL execution environment is typically installed on the home directory of the user "spell" (i.e. `/home/spell`).

In the following it is assumed that all the SPELL software is installed in `/home/spell/SPELL`. The environment variable **`SPELL_HOME`** shall be defined and it shall point to the aforementioned folder.

2 Configuration

The configuration files consist in a set of XML files, located in dedicated subfolders in `$SPELL_HOME/config`. Within this folder several subdirectories can be found: **server**, **contexts**, **spell**, **gui** and/or **dev**, depending on the SPELL software packages installed.

2.1 The server subfolder

This folder includes the XML configuration file that is used by the server at startup. This file is the main one and links to other configuration files. The name of the file is typically "server.xml" although the name can be freely chosen.

A server configuration file is composed of the following elements listed hereafter:

The common section: allows to enable or disable the server logging (Log) and the level of verbosity (LogDetail). The values accepted by the LogDetail tag are: PROC, MAIN, CNFG and COMM.

```
<common>
    <property name="Log">ENABLED</property>
    <property name="LogDetail">PROC</property>
</common>
```

The language section: it has one only element named “language” that provides the path to the language.xml file (see the “spell subfolder” section above).

```
<language>
    <file>language.xml</file>
</language>
```

The listener section: the listener is the component to which all SPELL client applications (e.g. SPEL-GUI) initially connect to. This section provides configuration parameters for the listener process. It accepts two property elements:

- the `ListenerPort` element to define the port to listen to
- the `ContextListenerPort` is the internal port used between the listener process and the context process.

```
<listener>
    <property name="ListenerPort">9980</property>
    <property name="ContextListenerPort">9981</property>
</listener>
```

The context section: the context process is configured with the parameters provided in this section:

- the `ExecutorProgram` element defines the location of the SPELL executor process binary.

```
<context>
    <property name="ExecutorProgram">$SPELL_HOME/bin/SPELL-Executor</property>
</context>
```

The executor section: the executor is the component that creates and manages the procedure instance at runtime. It supports the following parameters:

- **ExecDelay** (in milliseconds): the time to wait between the execution of two procedure lines of code.
- **PromptWarningDelay** (in seconds): the time to wait between a prompt appears and the injection of a warning event in the GCS, if the prompt does not get an answer.
- **RunInto** (True/False): Enables or disables the “Run Into” feature by default. This makes visible the execution of code within the functions or subprocedures themselves.
- **ByStep** (True/False): Enables or disables the “By Step” feature by default. This makes the execution pause at each line executing a Step statement.
- **BrowsableLib** (True/False): Allows or not to run into the UserLib source code at execution time.
- **MaxVerbosity** (integer): Defines the upper limit of verbosity to show at runtime. For more information, please refer to the Verbosity feature in the SPELL language manual.
- **WatchVariables** (ENABLED/DISABLED): Enables or disables the Watch Variables feature in SPEL-GUI. Mainly used for performance reasons.

```
<executor>
    <property name="ExecDelay">40</property>
    <property name="PromptWarningDelay">60</property>
    <property name="RunInto">False</property>
    <property name="ByStep">False</property>
    <property name="BrowsableLib">True</property>
    <property name="MaxVerbosity">5</property>
    <property name="WatchVariables">ENABLED</property>
</executor>
```

The drivers section: lists the available drivers in the system. The driver elements provide the configuration file name.

```
<drivers>
    <driver>driver_std.xml</driver>
    <driver>driver_xxx.xml</driver>
</drivers>
```

The available contexts section: provides the list of available contexts available for the server. The context element provides the configuration file name.

```
<contexts>
    <context>context_SAT1.xml</context>
    <context>context_SAT2.xml</context>
    <context>context_SAT3.xml</context>
</contexts>
```

The families section: maps the spacecraft name to the different GCS instances names. This allows identifying simulated instances to the real spacecraft, in order not to filter out the matching procedures (referring to the SPELL coding guidelines for more information on the procedures headers).

```
<families>
    <family name="FAM">SAT1,SAT2,SAT3</family>
</families>
```

2.2 The spell subfolder

The first (mandatory) file it includes is the language.xml file. It is used to redefine the defaults modifiers for each of the wrappers (named “function”) and core adapter objects (named “interface”).

For instance, the defaults modifiers definition for the TM object may look like the following:

```
    <defaults interface="TM">
        <modifier name="OnFailure">ABORT|SKIP|RECHECK|CANCEL</modifier>
        <modifier name="PromptFailure">True</modifier>
        <modifier name="ValueFormat">ENG</modifier>
        <modifier name="Timeout">30</modifier>
        <modifier name="Retries">3</modifier>
        <modifier name="Tolerance">0</modifier>
        <modifier name="Wait">False</modifier>
        <modifier name="Delay">None</modifier>
        <modifier name="PromptUser">True</modifier>
        <modifier name="OnFalse">NOACTION</modifier>
        <modifier name="OnTrue">NOACTION</modifier>
        <modifier name="IgnoreCase">False</modifier>
        <modifier name="AdjLimits">False</modifier>
    </defaults>
```

The list of modifiers depends on the interface or function that is being defined. Please refer to the SPELL language reference manual for the complete list of modifiers.

Other XML files available in this folder are named **driver_<drivername>.xml**. These files provide the configurations for each SPELL driver installed. The typical example is the driver_std.xml file corresponding to the standalone driver.

A typical content for this driver is:

```
<driver id="standalone">
    <name>Standalone SPELL driver</name>
    <interfaces>TM,TC,EV,TIME,TASK,RSC</interfaces>
    <maxproc>5</maxproc>
    <path>$SPELL_HOME/drivers</path>
    <properties>
        <property name="SimPath">Simulator/</property>
    </properties>
</driver>
```

The tags have the following meaning:

- **driver id**: a unique identifier that will be used to reference the driver in the server configuration file (refer to next section).
- **name**: a meaningful name for the driver.
- **interfaces**: allows to enable or disable the different interfaces
- **maxproc**: indicates the maximum number of procedures that can run in parallel per context.
- **path**: the path to the folder that contains the driver's specific binaries.
- **properties**: a free list of properties relevant to the driver. In the case of the standalone driver, the SimPath property indicates the subfolder name that contains the simulator data definition.

Finally, a configuration file for Log4CPLUS is included, "log.properties". An example of the contents of this file follows.

```
log4cplus.rootLogger=DEBUG, STDOUT

log4cplus.appender.STDOUT=log4cplus::ConsoleAppender
log4cplus.appender.STDOUT.layout=log4cplus::PatternLayout
log4cplus.appender.STDOUT.layout.ConversionPattern=[ %-5p ] [ %h ] [ %30.30l ]: %m%n
```

2.3 The contexts subfolder

This folder includes the configuration files that can be read at SPEL-Server startup. Refer to the This one links to the other XML configuration files.

A server configuration file is composed of the following elements listed hereafter. Note that the context id tag is a unique identifier for the context definition.

First appears a set of single elements:

- **name**: a label describing the context in short.
- **driver**: provides the driver to use for this context. See the spell subfolder description above for more information.
- **spacecraft**: gives the spacecraft family identifier for the context.
- **satname**: provides the identifier for the satellite instance.
- **gcs**: provides the hostname of the GCS to bind to.
- **procpath**: provides the path to the procedures base folder.

- **libpath**: provides the path to the User Library folder.
- **output_dir**: provides the base path for the files generated by procedures
- **input_dir**: provides the base path for the files read by procedures

```
<name>Standalone instance</name>
<description>Context for the Standalone simulator</description>
<driver>std</driver>
<spacecraft>STD</spacecraft>
<satname>STD1</satname>
<gcs>gcsserver</gcs>
<procpath>$SPELL_DATA/Procedures </procpath>
<libpath>$SPELL_DATA/UserLib</libpath>
<output_dir>$SPELL_DATA/OutputFiles</output_dir>
<input_dir>$SPELL_DATA/InputFiles</input_dir>
```

The driverconfig section: provides driver configuration parameters that apply to all procedures running for the spacecraft associated to the context. The example below sets up the parameters for a connection to a CORBA naming service, assuming the SPELL driver needs this kind of information.

```
<driverconfig>
    <property name="NameID">NameServiceId</property>
    <property name="NameServer">gcsserver</property>
    <property name="NamePort">29999</property>
</driverconfig>
```

The executor section: overrides the server configuration default parameters for the executor component, giving specific parameters for the associated spacecraft. See the server subfolder above for more information.

```
<executor>
    <property name="ExecDelay">100</property>
    <property name="RunInto">True</property>
    <property name="ByStep">False</property>
    <property name="BrowsableLib">False</property>
    <property name="MaxVerbosity">5</property>
    <property name="WatchVariables">ENABLED</property>
</executor>
```

The databases section: defines the parameters of the different URL prefixes as well as the preloaded databases. In the example below the SCDB and GDB global variables will exist and respectively contain the spacecraft database and the ground database.

The location element defines the path to the database folder in case of a filesystem database (relative to the SPELL_DATA environment variable) supports the following attributes:

- **name**: the unique identifier for the location
- **type**: may be file for regular file, asrun for the “As Run” files and svn for the revision controlled files.
- **ext**: the file name extension in case of a filesystem regular file.

The database element provides the database name (file name in case of filesystem based database) and supports the following tags:

- **name**: the name of the global variable that will contain the database and that will be provided to the running procedure.
- **location**: the location identifier (see above).

```
<databases>
  <location name="SCDB" type="file" ext="DB">InputFiles/SAT_SIM/SpacecraftDB</location>
  <location name="GDB"  type="file" ext="DB" >InputFiles/SAT SIM/GroundDB</location>
  <location name="mmd"  type="file" ext="IMP">InputFiles/SAT_SIM/ManeuvreMessage</location>
  <location name="usr"  type="file" ext="IMP">InputFiles/SAT_SIM/User</location>
  <location name="ar"   type="asrun" ext="ASRUN">OutputFiles/SAT SIM/AsRun</location>
  <location name="ws"   type="file" ext="WS">OutputFiles/SAT_SIM/Warmstart</location>
  <location name="svn"  type="svn" ext="SVN">InputFiles/SAT_SIM/User</location>

  <database name="SCDB" location="SCDB">SCDB</database>
  <database name="GDB"  location="GDB">GDB</database>
</databases>
```

The language section: allows to override the defaults of the functions and core adapter objects modifiers (refer to the spell subfolder section above).

```
<language>
</language>
```

3 Starting Up The SPEL-Server

The startup scripts for the SPELL applications are located in the $SPELL_HOME/bin directory. The server launcher script has the following syntax:

```
(2) ./SPELL-Server <Context Id>
```

It will start a SPELL Server using the the default server configuration file named "server_<Context Id>.xml", that is $SPELL_HOME/config/server/server_SAT.xml if the context identifier is "SAT".

When started, the server standard output displays some general information and log messages and goes in idle mode. The SPELL listener process is then ready to accept client connections.