Space Details

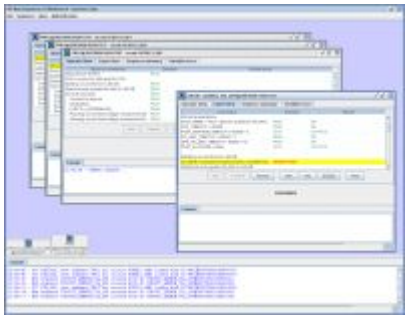
Key:	SEQ
Name:	Sequencer
Description:	
Creator (Creation Date):	administrator (Feb 19, 2007)
Last Modifier (Mod. Date):	administrator (Feb 19, 2007)

Available Pages

- Home
  - General Purpose Sequencer
    - Developers Info
    - Sequencer - Operator's Manual
  - HWC Sequencer
    - Deployment
    - Script Rules

Home

This page last changed on Aug 13, 2007 by daniele raffo.



This Wiki describes different aspects of the General Purpose Sequencer, as well as specific implementations and plugins for the HWC Sequencer, the LHC Sequencer and the SPS Sequencer.

The software can be found in the product repository, respectively: seq-app-gui seq-app-hwc seq-app-lhc seq-app-sps

You can report a bug, request a new feature or suggest an improvement on the AB/CO Issue Tracker.

General Purpose Sequencer

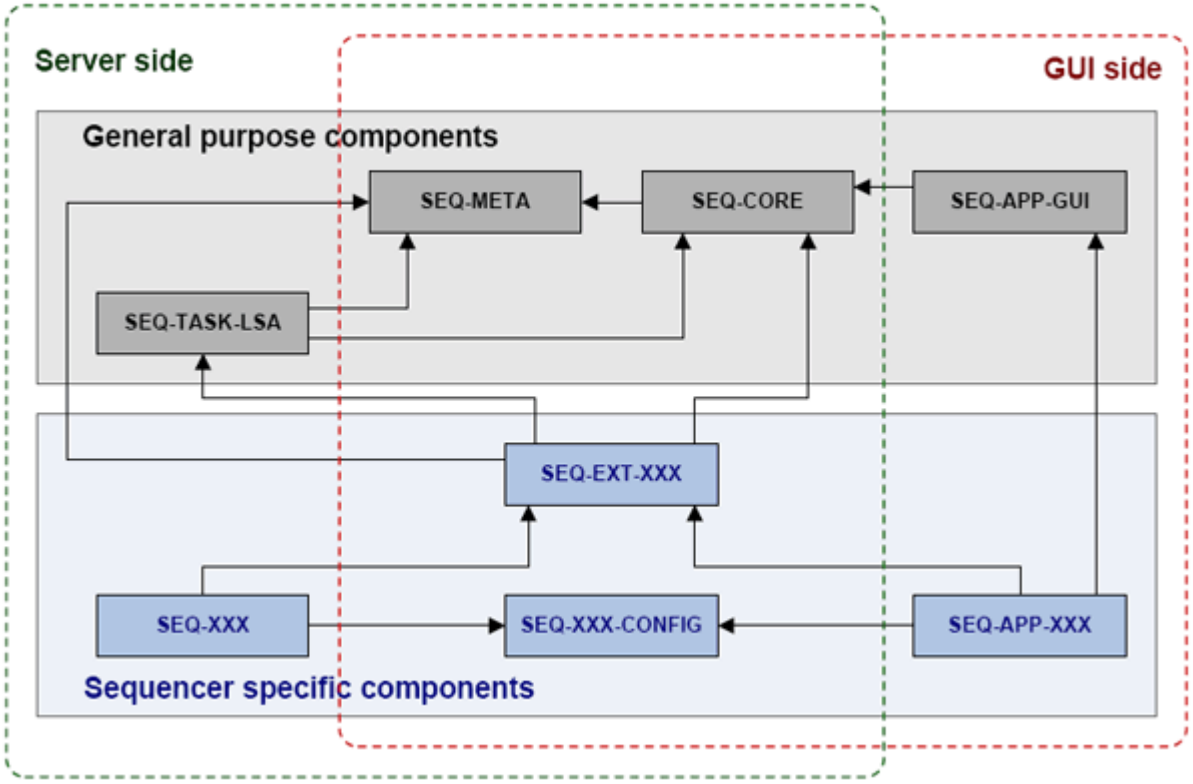
This page last changed on Aug 13, 2007 by daniele raffo.

This is the main page for the General Purpose Sequencer. It contains information that applies to all Sequencers: HWC, LHC, and SPS.

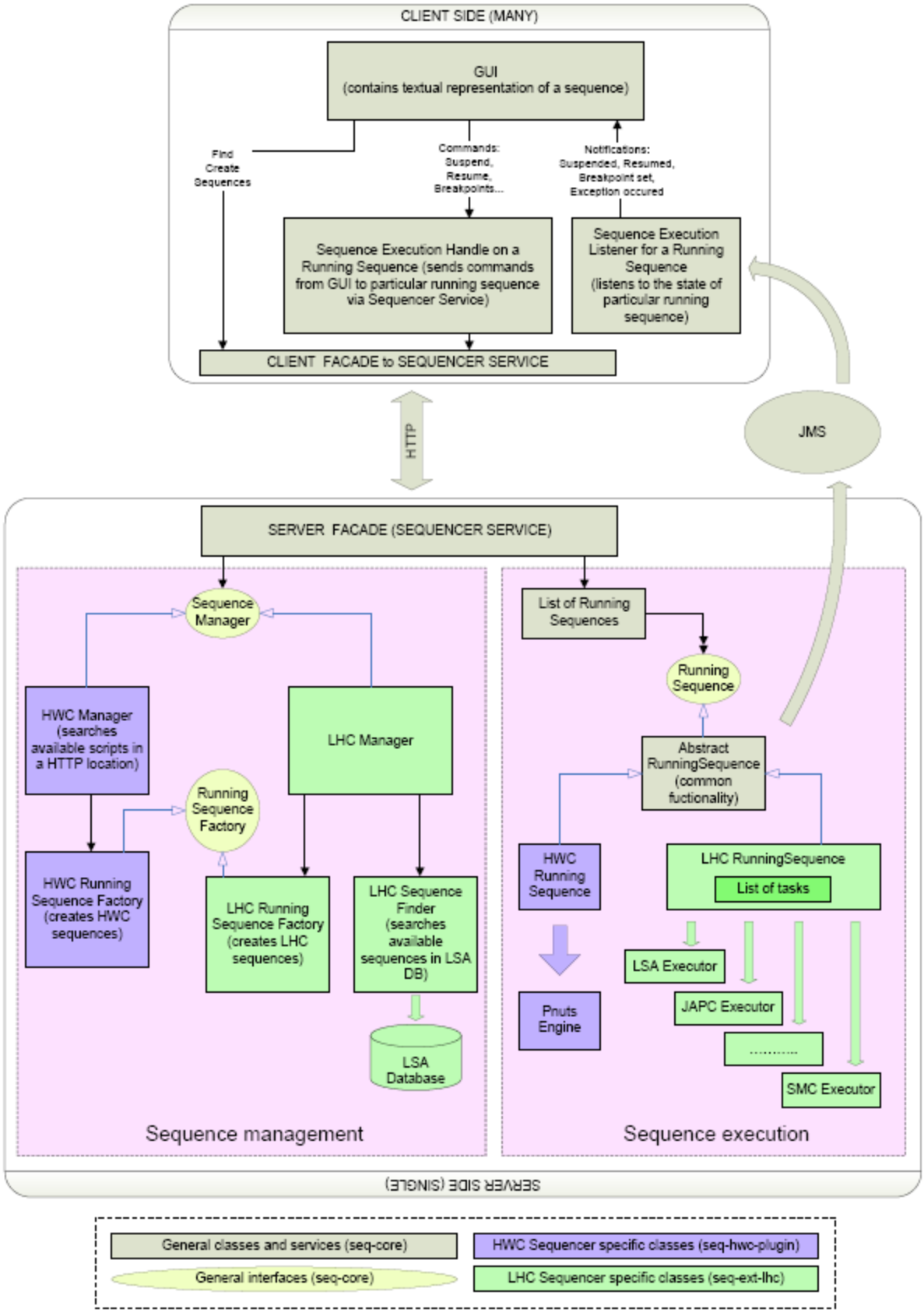
Developers Info

This page last changed on Aug 17, 2007 by daniele raffo.

Sequencer packages diagram



Schema of the Sequencer

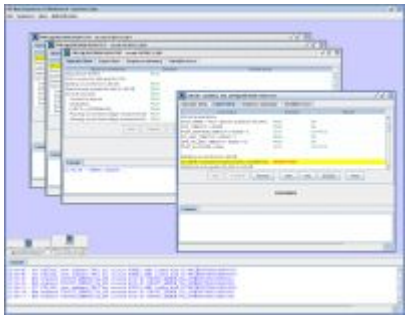


List of documents in this section:

	Name	Size	Creator	Date	Comment	
	Sequencer-schema.png	76 kb	Daniele Raffo	Aug 17, 2007	Overall Sequencer Diagram	Edit | Remove
	Sequencer-schema.pdf	44 kb	Roman Gorbonosov	May 24, 2007	Overall Sequencer Diagram	Edit | Remove
	SequencerPackageDiagram.gif	28 kb	Roman Gorbonosov	Aug 03, 2007	Sequencer Package Diagram	Edit | Remove
	SequencerPackageDiagram.pdf	18 kb	Roman Gorbonosov	Aug 03, 2007	Sequencer Package Diagram	Edit | Remove

Sequencer - Operator's Manual

This page last changed on Aug 17, 2007 by daniele raffo.



This is the Manual for the CCC Operators of the New Sequencer GUI.
The Sequencer can be used both with Hardware Commissioning (HWC), Large Hadron Collider (LHC), and Super Proton Synchrotron (SPS) facilities.

The Sequencer workbench

The title bar of the Sequencer shows:

1. The name of the Sequencer with the current version number
2. The facility on which the Sequencer operates: HWC, LHC, or SPS
3. The server on which the Sequencer is connected: PROD (Production) or DEV (Development)
4. The safety mode: SAFE or MD (see *Safety mode* below for an explanation)
5. The name of the operator currently logged in

Loading a sequence

You can load a sequence from the **File** menu. It offers the following choices:

- **Load a new sequence** - Loads a new sequence from the filesystem (HWC) or the database (LHC, SPS).
- **Reload last new sequence** - Reloads the same new sequence last loaded.
- **Observe a running sequence** - Shows a list of running sequences that you can load for observation. A running sequence is a sequence that was previously loaded on the server, from any operator, from any console. See below for an explanation of observing a sequence. Sequences enter the "running sequence" list when they are loaded, and are removed from the list 15 minutes after their termination.
- **Control a running sequence** - Shows a list of running sequences (as in the previous point) that you can try to take control on. If the GUI that loaded a new sequence crashes or is closed by the operator, the control on that sequence is released 45 seconds after this event and then the sequence can be controlled again by another GUI. The sequence's current point of execution, breakpoints and skip points are saved with the sequence itself.
- **Load a SOC test** *(HWC only)* - Opens the Set Of Circuits for Hardware Commissioning dialog that allows you to select a SOC, a circuit, and a test.
  In order to run a test, the relevant SOC must be reserved by you and the relevant circuit must be

unlocked; this is shown in green in the dialog.
You can select multiple circuits; in this case the tests list shows only those tests that are common to all selected circuits. Selecting a test and clicking OK starts the tests in parallel, opening a sequence window for each selected circuit. When all parallel sequences are finished or have been aborted, the SOC dialog pops up again.

The menu offers also commands that permit to update the latest version of a sequence from CVS. **File -> Refresh some sequence** allows you to select which sequence needs to be refreshed, while **File -> Refresh all sequences** refreshes all available sequences. This is useful if a programmer has modified a sequence and uploaded it on CVS. Sequences that are already opened are not refreshed.

All actions on sequences, except observing a running sequence, require you to authenticate by entering your operator login and password (currently none). This is *not* the same as your CERN NICE login/password.
You can authenticate and change operator login at any time via the menu item **Authentication -> Login**.
The name of the operator that is currently logged in is shown in the title bar of the Sequencer.

Executing a sequence

When a sequence is loaded, it is shown in a window that you can move, resize, iconify, and close, via the usual buttons in the top right corner.
The window title bar shows the ID of the sequence, as *sequence_name*@*datetime_of_creation*.
You can enlarge the window to the whole workbench width, or restore it to its original size, also by double-clicking the title bar.
To de-iconify an iconified sequence, double click on its icon.
Note that a sequence continues running even if you iconified or closed its window.

The top of the window shows one or more tab that can be selected.

- **Operator View** *(HWC only)* - This is the default view of the sequence table.
- **Expert View** - Shows the sequence table as a list of its atomic commands, one per line. This is useful for sequence programmers.
- **Sequence summary** *(HWC only)* - Shows some information regarding the sequence, i.e. ID and arguments with which it was called.
- **Variables trace** *(HWC only)* - Shows a tracing of the variables used in the sequence. Click on the **Refresh** button to refresh the list of names and values.

The sequence table is divided in three columns:

- The name of the command
- The directive to apply to the command: here you can choose from
  - RUN (default) to execute the command
  - BREAK to set a breakpoint on that command; in this case the execution will be suspended at that command line (the command will not be executed)
  - SKIP to skip the command
- The result of the execution of the command, which may be
  - OK if the command succeeded
  - FAILED if the command failed for some reason (followed by the error that caused the failure, e.g. a Java exception)

    - SKIPPED if the command was not executed because of a SKIP directive applied to it, but execution passed through that line anyway

Directives are automatically synchronized between Operator View and Expert View.
Note that you can apply a directive to multiple commands. To do this, select the commands, right click with the mouse button, and choose the appropriate directive.

Below you have the button panel to control the sequence. Only some button is clickable at any time, depending on the state of the sequence.

- **Start** - Starts the sequence, which will automatically run up to the end or until the first breakpoint.
- **Suspend** - Stops the sequence.
- **Resume** - Resumes the sequence, which will automatically run up to the end or until the next breakpoint.
- **Step** - Executes only the next command of the sequence. This button can be operated only when the sequence is suspended.
- **Skip** - Skips the next command of the sequence. This button can be operated only when the sequence is suspended.
- **Execute** *(HWC and SPS only)* - Executes a standalone command. Select a line, then click this button to execute the chosen command. This function is available only in Machine Development mode, and if the sequence is not started yet, is suspended or is finished.
- **Abort** - Aborts the sequence. You will be asked whether you want to wait for the current command to finish its execution, or to force an immediate abort.

If you are in Machine Development mode, you can enable all control buttons by right clicking on the table and selecting **Enable all control buttons**, thus overriding the sequence state. This is intended to be used in case of emergency only (e.g. GUI crashes or becomes irresponsive), therefore please choose this option wisely.

A yellow line in the table shows the command that is going to be executed next.
If you select the menu option **View -> Highlight last executed command**, a light yellow line highlights the command that was the last to be executed.
An orange line shows a standalone command that is being executed.

Dialogs

During the execution of a sequence, dialogs may appear. These dialogs require intervention from the operator.

When you click on a sequence window, the relevant opened dialog (if any) pops up in the foreground and its message is highlighted in yellow. When you iconify/de-iconify a sequence window, the relevant opened dialog (if any) is iconified/de-iconified in sync with its sequence window. (An iconified dialog is shown as a small square with a question mark inside.) This helps in telling which dialog purports to which sequence, if more than one sequence with dialogs is opened at the same time.

Safety mode

The Sequencer may run in two safety modes: Safe or Machine Development. The safety mode currently selected is shown in the Sequencer title bar.

The Safe mode prevents an operator from doing things that could potentially make the equipment unsafe.
To switch between the two modes, select **Authentication -> Switch safety mode**. You will be asked for the password if you are trying to go in MD mode.

In Safe mode, the following functionalities are disabled:

- Skipping commands (via the SKIP directive or the **Skip** button)
- Executing standalone commands (via the **Execute** button)
- Enabling all control buttons

Observing a sequence

In this mode, you merely observe the execution of a sequence that was recently loaded. The sequence window interface is similar as explained before, except for the fact that you don't own control over the sequence, and as such there aren't control buttons and you cannot change directives. However, you can still see in real-time which line is currently executed, the result of the lines, and the directives as they are changed by the operator that is controlling the sequence.

Working with sequence windows

Sometimes (for instance during a SOC test) you open many sequences in parallel. By default, these sequences are opened in cascade. If you want to open them side by side along the full screen's width, select **View -> Open sequence windows tiled in full width** before opening the sequences.

When you have multiple sequences at the same time, and the workbench is cluttered of sequence windows, you can use the **View** menu to organize them.
View -> Iconify all sequence windows iconifies all sequence windows so you can choose which one to reopen.
View -> Tile opened sequence windows reorganizes the sequence windows so that they are tiled side by side. This is done only on windows that are opened (non-iconified).
View -> Cascade opened sequence windows reorganizes the sequence windows so that they are put in cascade. This is done only on windows that are opened (non-iconified).

Version history

Only the latest releases are shown.

0.3.16	June 21st, 2007	Improved sequence dialogs: dialogs are highlighted and (de)iconified according to the parent sequence window.
0.3.17	June 27th, 2007	Added support for sequence refresh.
0.4.2 (NEXT)	July 26th, 2007	Project splitting for working with HWC or LHC facility. Directives

		can now be easily set on multiple lines.
0.4.3	to be scheduled	Added support for SPS facility. Added online help. Removed the **Sequence** menu for sake of simplicity.

-- Daniele Raffo AB/CO/AP

HWC Sequencer

This page last changed on Jul 26, 2007 by daniele raffo.

This is the main page for the Hardware Commissioning Sequencer.

This page last changed on Aug 17, 2007 by daniele raffo.

HWC Sequencer Architecture and Deployment

This page describes the architecture of the HWC Sequencer and its practical deployment. It includes instructions to identify and restart the corresponding processes.

Architecture

HWC Sequencer is deployed as a three tier architecture with many GUIs, a J2EE server and different execution agents:

- external systems, like LSA;
- direct access to Front-end computers.

Deployment (how to restart the HWC Sequencer)

We have two versions of the server: a production version and a development version.
The users and OP are connected to the production version, the development version normally runs the NEXT version to be deployed and is used by the development team.

Production Version

Here are the instructions to find the binaries and logfiles, and to restart the servers.
We have chosen a `wreboot` set-up that only restarts the process but does not copy over a new version from the repository.

These processes can be restarted using `wreboot`. A restart has no impact on the equipment (settings are not influenced) and the users will only notice a short interruption (60 seconds).

HWC Sequencer Production server

- Machine: `cs-ccr-lsa2`
- Directory `/local/seq_hwc_server/prod`
- Logfiles:
  - `/local/seq_hwc_server/prod/logs/seq-hwc-server.log` (20 backups are kept)
  - `/local/seq_hwc_server/prod/run/stdout.log` (owerwritten with every restart)
  - `/local/seq_hwc_server/prod/journal/` (journal files of the tests)
- Wreboot commmand: `wreboot -n SEQ_HWC`

ActiveMQ broker used in production

- Machine: `cs-ccr-spsea2`
- Directory `/local/jms-sequencer-pro/`
- Logfiles: `/local/jms-sequencer-pro/activemq-data/activemq.log`
- Wreboot commmand: `wreboot -n jms-sequencer-pro`

HWC Sequencer Production GUI

The production version of the HWC Sequencer GUI can be started from this JNLP URL. Please don't start it from a wireless computer because the low bandwidth may seriously slow down updates for all GUIs connected!

HWC Sequencer Development server

- Machine: `cs-ccr-lsa2`
- Directory `/local/seq_hwc_server/dev`
- Logfiles:
  - `/local/seq_hwc_server/dev/logs/seq-hwc-server.log` (20 backups are kept)
  - `/local/seq_hwc_server/dev/run/stdout.log` (owerwritten with every restart)
  - `/local/seq_hwc_server/dev/journal/` (journal files of the tests)
- Wreboot commmand: `wreboot -n SEQ_HWC_DEV`

ActiveMQ broker used in development

- Machine: `cs-ccr-spsea2`
- Directory `/local/jms-sequencer-dev/`
- Logfiles: `/local/jms-sequencer-dev/activemq-data/activemq.log`
- Wreboot commmand: `wreboot -n jms-sequencer-dev`

HWC Sequencer Development GUI

The development version of the HWC Sequencer GUI can be started from this JNLP URL. Please don't start it from a wireless computer because the low bandwidth may seriously slow down updates for all GUIs connected!

Script Rules

This page last changed on Aug 09, 2007 by daniele raffo.

Sequences for HWC are written using the Pnuts language, a Java-derived scripting language that runs on the JVM.

Script Rules

Sequencer comments

The HWC Sequencer includes a parser that analyzes the script and shows it in a user-friendlier way.

To use this feature, you can include *Sequencer comments* in the Pnuts script. These are lines starting with a *triple slash* (`///`), and are ignored by the Pnuts core engine (as they are valid Java comments). A Sequencer comment marks the beginning of a *block*, which ends to the script line immediately before the next Sequencer comment, or to the last script line.

In this way you can group script lines into a single block. Expert View shows the original script, one line per row, while Operator View shows the blocks, one block per row. The Operator View allows the operator to perform an action on multiple command lines: for instance, clicking on the **Step** or **Skip** command buttons while in Operator View will respectively step or skip through all script lines of the relevant block.

As an example, this is the `INITIALIZE_CIRCUIT.pnut` script. It is shown as such in Expert View:

```
//Circuit initialization
//This sequence needs 3 parameters: socName, circuitName, operatorName

///Script preparations
FGC_TIMEOUT = 60000;
QPS_OK_INIT_TIMEOUT = 10000;
TEST_SUCCESS = false;

///Selecting a circuit
socName = SeqDialogs.showSelectOneDialog("SOC name", "Please type the Set-of-circuits name:",
getOperationalSocNames());
if (socName == null) {
        return;
}
socInfo = new SocInfoImpl(socName);
circuitName = SeqDialogs.showSelectOneDialog("Circuit name", "Please select a circuit to
initialize:", socInfo.getCircuitNames
());
if (circuitName == null) {
        return;
}
circuitInfo = HwcUtil.prepareCircuit(socName, circuitName);

pic = null;
fgcStateCtrl = null;
try {   ///TRY
        ///Connecting to devices
        pic = circuitInfo.getPic();
        fgcStateCtrl = circuitInfo.getFgcStateCtrlAtomic();
        fgcFaultCtrl = circuitInfo.getFgcFaultCtrlAtomic();
        ///Circuit initialization
        ///Remove the PC_PERMIT at the PIC level.
        pic.signalInit();
```

```
        pic.removePcPermit();

        ///Set PCs to OFF and wait until they are OFF or FLT_OFF
        fgcStateCtrl.setState(FgcStates.OFF);
        fgcStateCtrl.waitForState(FGC_TIMEOUT, [FgcStates.OFF, FgcStates.FLT_OFF]);

        ///Check that only NO_PC_PERMIT present
        if (!fgcFaultCtrl.exactFaultsPresent([FgcFaultCodes.NO_PC_PERMIT])) {
                throw new Exception("Other faults present besides NO_PC_PERMIT on PCs: " +
fgcStateCtrl.getDevices());
        }

        ///Check QPS_OK
        if (!circuitInfo.isCType()) {
                pic.waitQpsOk(QPS_OK_INIT_TIMEOUT);
        }

        ///PIC gives PC_PERMIT for the circuit (send command to close according relay)
        pic.signalInit();

        ///Finalization
        TEST_SUCCESS = true;
} catch (Exception ex) {        ///CATCH
        ///Exceptional finalization
        exceptionDescription = SeqUtil.describeException(ex);
        System.out.println(exceptionDescription);
        SeqDialogs.showInfoDialog("Exception", exceptionDescription);
} finally {     ///FINALLY
        ///Message with result
        if (TEST_SUCCESS) {
                SeqDialogs.showInfoDialog("Init success!", "Init of circuit " + circuitName + "
finished successfuly ");
        }
        else {
                SeqDialogs.showInfoDialog("Init Failure!", "Init of circuit " + circuitName + "
failed");
        }
}
```

This script is then visualized as such in Operator View:

```
SEQUENCE START
Script preparations
Selecting a circuit
TRY
        Connecting to devices
        Circuit initialization
        Remove the PC_PERMIT at the PIC level.
        Set PCs to OFF and wait until they are OFF or FLT_OFF
        Check that only NO_PC_PERMIT present
        Check QPS_OK
        PIC gives PC_PERMIT for the circuit (send command to close according relay)
        Finalization
CATCH
        Exceptional finalization
FINALLY
        Message with result
```

Note that the parser inserts a first block named "SEQUENCE START" that includes the first script lines. Note also that a Sequencer comment can also be put at the end of a script line: in this case, the block includes that line.

The operator can as well set directives (RUN, SKIP and BREAKPOINT) on blocks, being Expert View and Operators View automatically synchronized.
In fact, in Operator View:

- setting a RUN on a block (in Operator View) will set RUN on all script lines (in Expert View) purporting to that block

- setting a SKIP on a block (in Operator View) will set SKIP on all script lines (in Expert View) purporting to that block
- setting a BREAKPOINT on a block (in Operator View) will set BREAKPOINT on the first executable script line (in Expert View) purporting to that block

and in Expert View:

- setting a RUN on all script lines purporting to a block (in Expert View) will set RUN on that block (in Operator View)
- setting a SKIP on all script lines purporting to a block (in Expert View) will set SKIP on that block (in Operator View)
- setting a BREAKPOINT on any script line (in Expert View) will set BREAKPOINT to the block (in Operator View) the script line purports to

Remember that the Sequencer will not allow you to set a directive on a non-executable script line, e.g. a comment or a blank line; this because that line is ignored by the Pnuts core engine. In the same manner, you cannot set a Directive on a non-executable block (a non-executable block is a block composed of all non-executable script lines). For instance, the SEQUENCE START block in the example is non-executable.

No simple blocks

In Sequencer scripts is prohibited to use *blocks of code alone*, without any including structure like if statements, loops, try/catch/finally constructs. So, the following code is illegal:

```
a = 5;
{
     b = 8;
}
c = 10;
```

But the following code is correct:

```
a = 5;
if (a == 5) {
    b = 8;
}
c = 10;
```

This requirement is needed because allowing simple blocks of code would prevent to correctly handle brutal cancellation of a sequence.

No thread pool usage

In Sequencer scripts is prohibited to use any kind of *thread pools*. This requirement comes from the fact that thread-local variables are used to keep the context information for each sequence.