SILABILITY USER GUIDE

A COMPLETE GUIDE TO VERSION 2.2 OF SILABILITY, A DESKTOP SOFTWARE PACKAGE FOR SIL VERIFICATION CALCULATIONS

DEVELOPED BY XSERICON

Table of Contents

For further information and support, contact xSeriCon via info@xsericon.world or call +852 9439 3200.

The SILability user guide was written by Peter Clarke and last updated in June 2021.

1. CHAPTER 1 - INTRODUCTION TO SILABILITY

1.1 WHAT IS SILABILITY?

Industrial facilities and equipment commonly use automatic safety functions to control risks. These functions are often known as Safety Instrumented Functions (SIFs) or Instrumented Protective Functions (IPFs). Their purpose is to monitor specific parameters (such as temperature of a tank, position of a moving part, speed of a motor, or the presence of a person or object using a sensor beam) and take action when given conditions are met (such as motor over speed).

SIFs are designed to reduce or limit the frequency of unwanted outcomes that may result from specific upset conditions. The design of each SIF should specify a numerical value representing the target performance of the SIF. This can be expressed in the following ways:

- Maximum probability of failure on demand of the SIF, averaged over its design lifetime (PFDavg)
- Minimum frequency reduction of the unwanted event—expressed as the ratio of the event frequency without the SIF, to the maximum tolerable frequency with the SIF; this is known as the SIF's target *risk reduction factor* (RRF)
- Maximum probability of failure per hour of the SIF (PFH); this measure is used for hazardous conditions that are always or frequently present

During the basic design phase of a SIF, the design team should confirm that the proposed SIF design is capable of achieving the target performance measure. This task, known as *SIL verification*, is a requirement of the relevant international standards (see chapter 7 for further details). SILability's purpose is to assist the designer in executing SIL verification correctly, accurately and efficiently.

1.2 WHAT DOES SILABILITY DO?

SILability performs calculations to confirm whether two of the specific requirements of the standards are met. While a full description of the requirements is beyond the scope of this User Guide, they can be outlined briefly as follows:

- *Random hardware failure performance requirement:* The predicted failure performance of the SIF in terms of random hardware failures must meet the specified target. For example, the SIF's RRF may need to exceed 25.
- *Architectural constraints requirement:* The SIF must meet a target level of hardware fault tolerance—that is, the number of hardware faults that can simultaneously occur in the SIF without impairing the intended function.

Calculating whether these requirements are met involves a large number of input parameters. SILability prompts the user to enter values for all required parameters, and provides guidance when necessary values are missing or inappropriate. Calculations are performed on the fly, so that the user can instantly see the effect of changing a parameter value.

Please note that the two requirements above are not the only requirements for demonstrating that a SIF achieves compliance with the standards. In particular, users should be aware of the *SIL capability requirement*: all the hardware and software used in a SIF needs supporting evidence to show that it is sufficiently free of design errors that could lead to systematic failures. SILability does not provide support for this requirement, which must be addressed and documented by other means.

The use of SILability assumes reasonable knowledge of the standards and the basic philosophy of SIL verification in the context of functional safety engineering. If you are not sure of the meaning of any of the functions or terms used in this user guide, please seek professional guidance from xSeriCon before use. For a quick introduction, please check out xSeriCon's videos on YouTube (search: xSeriCon youtube SIL verification).

1.3 WHAT IS THE PHILOSOPHY OF SILABILITY?

Because such a large number of input parameters are required for SIL verification, it is all too easy to enter wrong parameter values by mistake. For instance, a simple typing error, such as entering 1.62 instead of 16.2, could have a drastic effect on the result. Ultimately, this could lead to a dangerous level of under-protection provided by the as-built SIF.

In order to minimise the risk of incorrect parameter values, SILability has the following features:

- Default values are never provided. Any numerical field for which the user has not entered a value is displayed as *Not set*. No assumed values will be used in the calculation. (The exceptions are the SIL target field, which defaults to 4, the most conservative target; and the component AC type, which defaults conservatively to type B.)
- The user can enter numerical value in any appropriate units. For example, test intervals can be entered in days, weeks, months or years. This means the user does not have to do any conversion by hand. The selected unit is clearly displayed at all times.
- A large battery of 'reasonableness checks' is performed on the data on request. These are intended to spot data inconsistencies, values outside the typical range, and differences between SIFs that might be unintentional.
- A future version of SILability will allow users to link fields together so that, if one field is changed, its linked fields will be changed automatically. This allows the user to ensure consistency between items in the calculation. The user is made aware of the changes made to linked fields.

Many other features are also under planning to help ensure that input data is correct. Registered users of SILability will be informed of new releases as soon as they are available.

Traceability is extremely important in SIL verification. SILability supports this by providing a powerful comment feature. Almost every item of input data can have multiple comments attached to it. These comments can be used to indicate the source of the data, explain discrepancies, or for any other purpose. All comments* are shown in the final report.

All project data files generated by SILability are in a human-readable, portable format known as XML. This means, when you need to import SILability data into another application, it should be straightforward to write a data conversion tool, depending on the import capabilities of the other application. The designers of SILability chose this approach to help users minimise the risk of data getting changed inadvertently during export/import, and to reduce the need for retyping or copying/pasting data between applications. Details of SILability's XML format are given in Appendix D of this user guide.

A number of assumptions relating to the design of the SIF are made in the calculations. All assumptions are listed in Appendix B of this user guide.

1.4 FEATURES AND LICENSING

SILability is copyright-protected, licensed software. If you haven't yet purchased a license, you can view existing SILability data files and perform a limited range of tasks. The full features of SILability require you to have a purchased software key from xSeriCon; please see our contact details at the front of this User Guide.

1.5 HOW TO GET INFORMATION AND HELP

The designers of SILability are ready to help with any questions you may have. We are available to provide training on SILability and on the theory of SIL verification, via distance learning or in a classroom format; please contact us for details of availability and pricing.

We welcome your feedback on SILability, including any problems you encounter while using the software, or suggestions for improvement. Our contact details are at the front of this User Guide. When sending us feedback or problem reports, please tell us:

- The version of SILability you are using (select 'About SILability' in the welcome screen or the File menu)
- The exact version of your operating system (e.g. MacOS 11.1)
- What command or function of SILability you were using when the problem arose (please be as specific as possible), and which SIF you were working on
- Whether you have been able to repeat the problem

* Certain exceptions are noted in the FAQ; see Appendix H.

- Please send us the SILability project and database files you were using; we will handle your data in strict confidence and use the data only for purposes of resolving the issue you reported.
- Please look in your home folder (e.g. “My documents”) for a file named SILabilityError.txt, and send it to us. This contains useful debugging information to help us determine what went wrong.

2. CHAPTER 2 - SETTING UP SILABILITY

2.1 INTRODUCTION

This chapter describes how to set up SILability on your computer, and covers licensing arrangements.

SILability requires a computer (desktop, laptop or tablet) running Windows 8.1 or later, or OS X/MacOS. An internet connection is required for license checking each time you start SILability. (See section 2.7 for what to do if you don’t have internet access.)

2.2 GETTING SILABILITY

On request, we will send you a link to download the latest release of SILability. For your safety, please virus-scan the downloaded package before using it.

2.3 INSTALLING SILABILITY

On Windows, the download file is named ‘Install.exe.zip’. Run this file by double-clicking it (you may need administrator rights for this). SILability will install automatically.

Windows requirements: - Stable Internet connection

- Minimum screen resolution 1152 x 864

- Windows 8.1 or later

On MacOS, the download file is named ‘SILability’. Run this file by double-clicking it (you may need administrator rights for this). SILability will install automatically.

Mac requirements: - Stable Internet connection

- OS X/MacOS Yosemite v10.10.5 or later

2.4 GETTING SOFTWARE UPDATES

If your email address is registered with xSeriCon, we will notify you when a new version is available. You can download and install the latest version from the link we provide. Your existing license will remain valid within the same major version number (e.g. within version 2, including 2.0, 2.1, 2.2…). To benefit from an upgrade to a new major version number, you will need to request a replacement license.

2.5 LICENSING

SILability is licensed on a per-machine, per-version basis. For instance, if Susan purchases a license and uses it on her laptop for SILability 2.0, it will only work on Susan's laptop, and only for SILability 2.x (every version starting with the same major version number).

You need to purchase a license from xSeriCon to gain access to all of SILability's features. You can also request a trial license free of charge for a limited period. However, even if you don't have a license, you can try out many of SILability's functions.

The following functions depend on the type of license you have:

	Trial	Full
Saving a project file	✓	✓
Creating a new project with 'save on the fly' enabled	✓	✓
Add a new SIF to a project		✓
Add multiple components, legs or groups to a SIF	✓	✓
Copy a SIF		✓
Generate Excel output from a project	✓	✓

If you uninstall and reinstall SILability on the same computer (within the same major version number), your existing license key will still work. You can enter the same key again.

If you want to transfer your license to another computer, please contact xSeriCon. We will issue you with a new license key, with the same expiry date as your old license, and cancel your existing license key.

If you update your OS, especially on Mac, SILability may not be able to recognise your computer any longer. Please contact xSeriCon and we will immediately issue a new license

for you, at no extra charge. If you are planning to update your OS, contact us beforehand and we will give you a new license key, which you can start using immediately after your update.

For more guidance on licensing, please refer to Appendix H in this user guide.

2.6 RUNNING SILABILITY

Make sure your computer has internet access when starting SILability. This is needed to check the license. If you don't have internet access, you can't use the functions listed in section 2.5. (However, see section 2.7 for what to do if you don't have internet access.)

On Windows, launch SILability by doing one of the following:

- Select "SILability" in the Start menu.
- Double-click the "SILability" icon on the desktop.
- Search for "SILability" (press windows key + 'S' and type SILability)

On macOS, launch SILability by doing one of the following:

- Double-click the "SILability" icon on the desktop, or single click in the dock or launchpad.
- Double-click the "SILability" icon from the "applications" folder within the "Finder" tool.
- Search for "SILability" using the Spotlight search tool.

You will see SILability's welcome screen. When first launched, it will show that SILability is unlicensed. If you have a license key, click "Enter license key", type or paste the license key you received from xSeriCon, and press <Enter>. You will see a confirmation message indicating whether the license key was accepted.

You only need to enter the license key once; SILability will store your license key automatically, if it's valid. On subsequent runs, SILability will use the stored license key. If you get a new license key (e.g. for an updated version of SILability or after your key has expired), click "Enter license key" and enter the new key, following the same procedure as the first time.

Now you can start using SILability. See chapter 3 for how to get started with your SIL verification project.

In Windows, you can also launch SILability and open an existing project by opening the project file (e.g. double-click on the project file in your file manager). This will open the project with 'save on the fly' activated (see section 3.2.1 for details).

2.7 RUNNING SILABILITY WITH NO INTERNET ACCESS

If you don't have internet access, or there is a problem with xSeriCon's licensing server, you can contact xSeriCon for an emergency key. This will unlock SILability for a limited time, giving you access to all of its functions.

Enter the key in the same way as a normal license key (click "Enter license key" on the Welcome screen).

2.8 SELECTING A LANGUAGE

SILability can be run in various languages. To select a language, launch SILability and press the "Select language" button on the Welcome screen. Your language choice will be stored and applied each time you start SILability in future.

3. CHAPTER 3 - MAKING A PROJECT

3.1 INTRODUCTION

What is a 'project' in SILability? A project is a set of Safety Instrumented Functions (SIFs). SIFs are sometimes known as Instrumented Protective Functions (IPFs), trips, or interlocks. Chapter 4 provides a brief introduction to the concept of a SIF and its architecture.

The project also includes a set of descriptive parameters such as project name and date. These are described in the next section.

3.2 PROJECT PARAMETERS

The following parameters are used for reporting and traceability purposes. You can edit them by choosing File menu – Project settings.

Parameter name	Meaning
Project name	The project name that will be shown in the SIL verification report
Client	The owner of the risk

Version number	Version control number of the SIL verification task. You should increment this manually after every significant change to the data in the project.
Edit number	This number, generated by SILability, increments automatically after every change to a data field that is used in the calculation. Its purpose is to allow you to confirm that the current state of the data exactly matches the data used to produce the final report.
Team members	The names of personnel involved in the SIL verification task
Date initiated	The start date of the SIL verification task

3.2.1 Save on the fly

As you work on your project, SILability can save your work for you continuously. You don't need to use the 'Save' command. This concept is called 'save on the fly.'

Even if your work is interrupted by a computer crash or power loss, none of your work should be lost if 'Save on the fly' is active. You can simply re-open the project file and SILability will recover the project automatically.

To start Save on the fly, use the File menu – Save project command and check the "Keep saving on the fly in the new file" option.

If you want to make experimental changes to the data without saving them, you can use the Sandbox function (see section 5.8).

3.2.2 The welcome screen

The welcome screen is visible when you launch SILability, and also whenever no project is open.

From here, you can create a new project or open an existing project.

To return to the welcome screen, close your project in the main SIL verification screen using File menu – Close project.

3.3 CREATING A NEW PROJECT

Projects are created from templates. A SILability template file is simply an XML project file that resides in the Templates folder in SILability's workspace (the folder named SILability or .SILability as explained in Chapter 2).

To create a project, select “Start new project” from the welcome screen. On the next dialogue, select the template you want to use. If you want to save the new project on the fly, select “Make new project and save my work”. Otherwise, select “Let me try it out”; this option will create a project, but the project will not be saved until you select File menu – Save project in the main SIL verification screen.

You can add your own template just by copying a valid project file to the Templates folder. It will then show up in the templates list next time you select ‘Create new project’ from the welcome screen.

3.4 SAVING A PROJECT

Projects are saved automatically if ‘save on the fly’ is active. To activate save on the fly, or to save manually, select File menu – Save project command. This offers you the choice of saving once, or starting save on the fly (so that all future changes in this session are saved automatically).

To check whether save on the fly is active, select File menu – Project information.

3.5 CLOSING A PROJECT

The ‘close project’ command is in the File menu. A warning will be shown if there are unsaved changes. Also, if there is an open database with unsaved changes, a warning will be shown. When the project is successfully closed, the Welcome screen is shown, allowing you to open another project if you wish.

3.6 OPENING AN EXISTING PROJECT

The current version of SILability allows only one project open at a time. (Future versions will remove this limitation.) Any open project must be closed before you can open another one. You can open a project from the Welcome screen, using the ‘Open project’ or ‘Open recent project’ buttons, or by double clicking on the project file in your file manager.

When you first open the project, it will not be saved on the fly. (This is to prevent unintended changes to your saved project.) To start save-on-the-fly, use the Save As command from the File menu in the main SILability window. We strongly recommend that you always use save-on-the-fly whenever you intend to make permanent changes to your project.

4. CHAPTER 4 - WORKING WITH SIFs

4.1 INTRODUCTION

In Chapter 1, the concept of a SIF was introduced. This chapter explains how SIFs are handled in SILability, including how to create, select and delete SIFs and how to navigate around the structure of a SIF.

4.2 CREATING A SIF

When you start a new project in SILability using the “minimal” template (as described in Chapter 3), your project will contain one SIF with no data. You can start using this SIF immediately.

To create another SIF, go to the SIF menu and select one of the following:

- New SIF at end (adds a new SIF after the last SIF in the project)
- New SIF after this one (inserts a new SIF below the currently selected SIF)
- New SIF before this one (inserts a new SIF above the currently selected SIF)

The new SIF will be created and selected. If you decide you don’t want the new SIF, select Tools menu – Undo to go back to the previous state of the project.

4.3 SELECTING A SIF

When you want to view or work on a different SIF, select the SIF you want in the “Go to SIF” choice box at the top of the data entry panel.

4.4 DELETING A SIF

Select the SIF you want to delete (using the “Go to SIF” choice box). Select SIF menu – Delete SIF. If you delete a SIF by mistake, use the Undo command in the Tools menu.

You can also delete one or more SIFs from the SIF list. Select Tools menu – Show SIF list. See section 4.12 for details.

You can’t delete a SIF if it is the only SIF in the project, because each project must contain at least one SIF.

4.5 COPYING AND IMPORTING SIFS

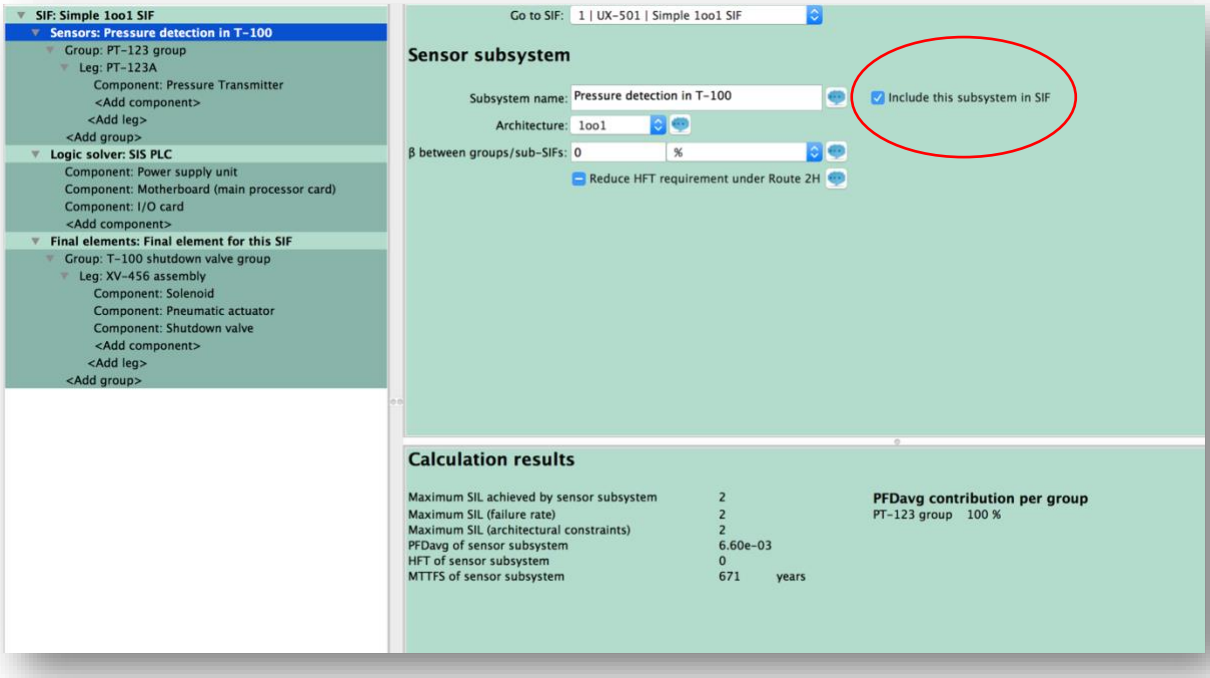
To copy a SIF, select SIF menu – 'Copy SIF,' or 'Copy SIF and go to the copy.' The new SIF is inserted directly below the original SIF. You can undo SIF copying if necessary.

4.6 ARCHITECTURE OF A SIF

SIFs consist of three *subsystems*:

- **Sensors:** the field devices typically connected to the equipment under control, including input devices such as sensors, transmitters, pushbuttons, signal conditioners, intrinsically safe (IS) barriers, and wiring up to the input of the logic solver.
- **Logic solver:** a device that decides whether the SIF should be in the tripped or untripped state, based on signals from the input devices. Typically, this is either a programmable logic controller (PLC) or an assembly of safety relays. It includes any needed ancillaries such as power supplies.
- **Final elements:** the field devices that act on the equipment under control, in order to achieve the design intent of the SIF. Typical examples are valves, motor control circuits, clutches, relays, and solenoids. All equipment whose correct functioning is important to achieve the safe state of the equipment should be included, starting from the output of the logic solver.

SILability allows you to model secondary SIFs (having no sensor subsystem) or partial SIFs (with one or two empty subsystems empty). To achieve this, a checkbox is provided, "Include this subsystem in SIF", to allow empty subsystems to be ignored.



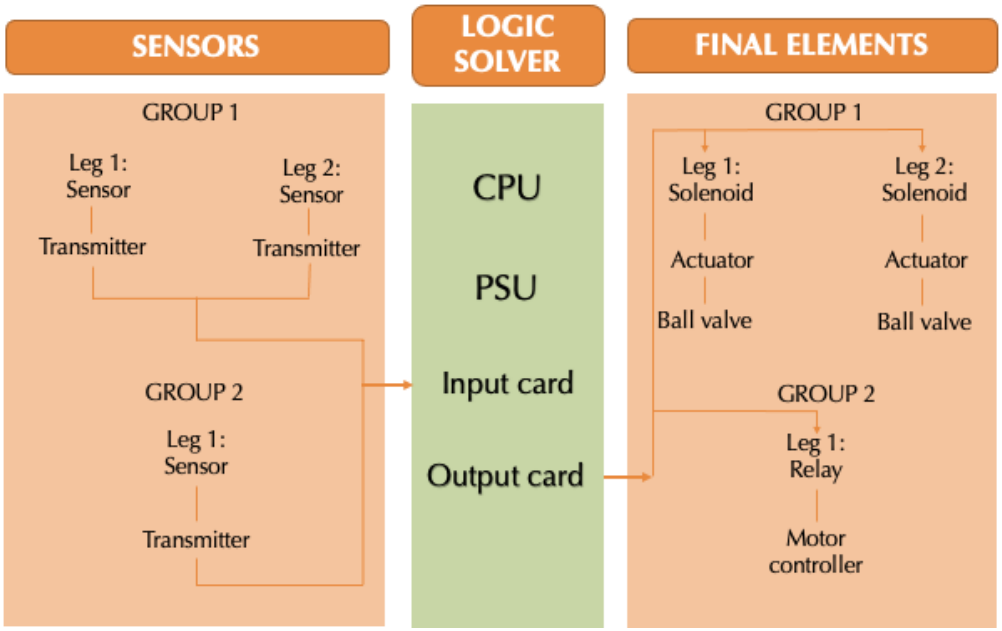
Field devices in the sensor and final element subsystems may be numerous, and structured in a complex logical relationship. To allow this to be accurately modelled, SILability provides three levels of hierarchy: *groups*, *legs* and *components*.

Individual components such as sensors and transmitters are assembled into legs; within a leg, the components are assumed to be logically connected in series, such that all components must function correctly for the leg to function as a whole (i.e. *N*oo*N* architecture). Each component is assumed to have no internal redundancy (Hardware Fault Tolerance, HFT = 0); however, SILability can handle components such as redundant solenoids with HFT = 1. See section 6.7 for details.

Legs are assembled into groups. Sometimes, each leg in a group will be identical; for example, in a 2oo3 overpressure SIF, three legs will be provided, each containing a pressure sensor, pressure transmitter and perhaps an IS barrier. However, in 1oo2, 1oo3, 1oo4, 2oo2, 3oo3 and 4oo4 architectures, SILability allows the legs to be non-identical if required.

Groups allow you to model situations where the sensors have multiple ways to detect the hazard. For example, a distillation column protection SIF may trip on high pressure or high temperature, both arising from the same causes. To model this, put the pressure sensor legs in one group, and the temperature sensor legs in another. Similarly, groups can also be used in final element subsystems, for cases where the SIF takes multiple actions (in a 1oo*N* or *N*oo*N* logical relationship) to achieve the safe state.

Schematic diagram of example SIF structure as modelled by SILability



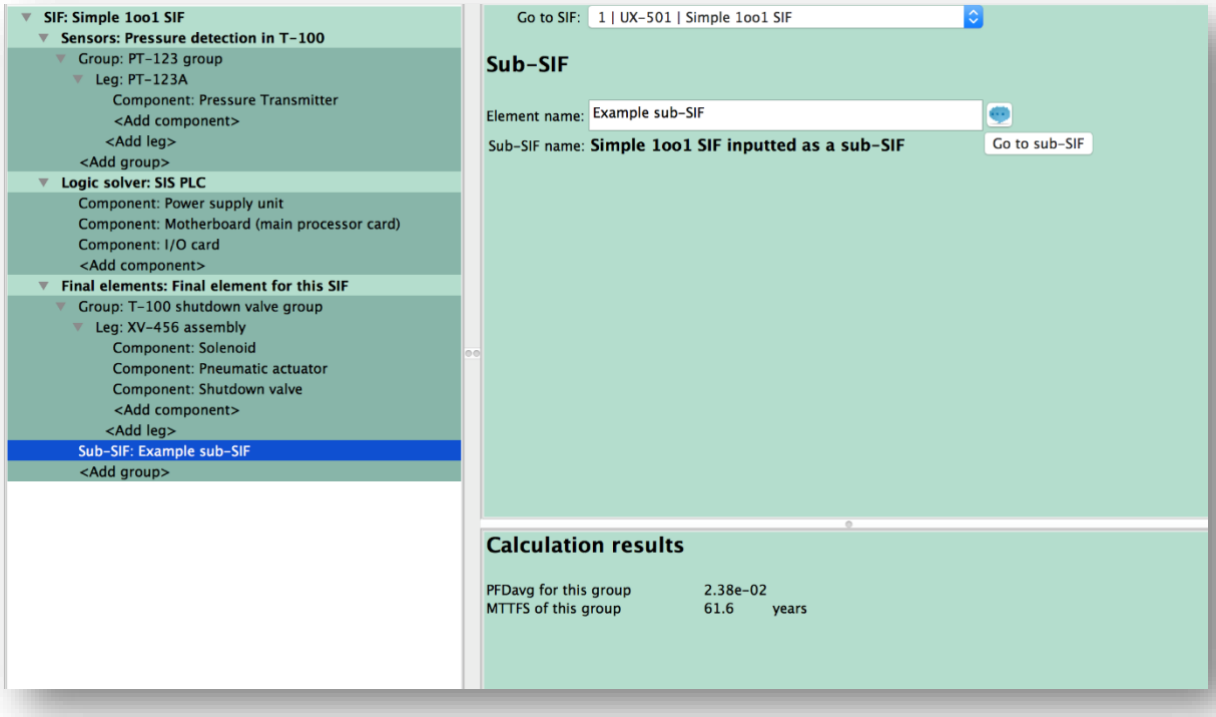
The internal logical architecture of the logic solver (from the SIL verification point of view) is usually more straightforward, so groups and legs are not provided. Instead, you can list all the main components of the logic solver directly in the subsystem, such as I/O cards, CPU,

power supply, and relays. The extent to which you subdivide the logic solver's components will depend on the granularity of the failure rate data available.

4.7 SUB-SIFS

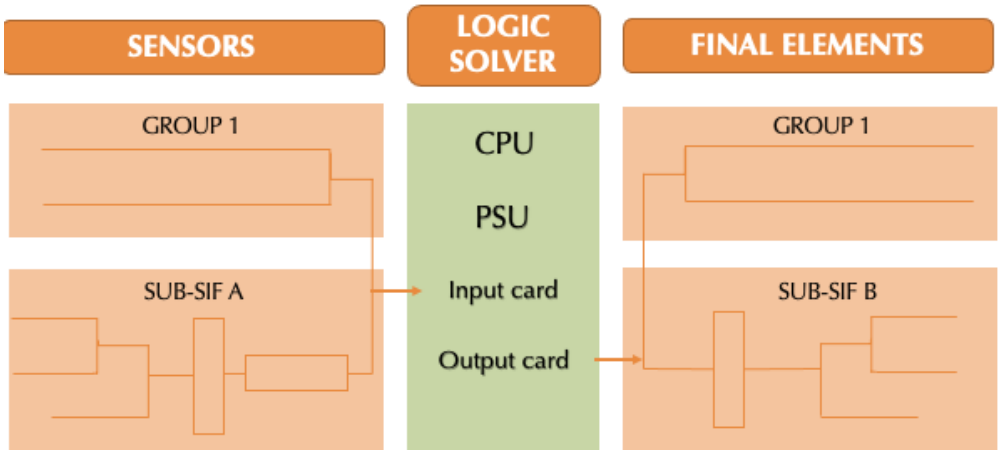
Each SIF can include another SIF as part of its architecture. This enables you to construct SIFs of complex architecture, and provides a method of inserting complete subassemblies (such as groups of sensors, groups of valves, and machine monitoring systems) into multiple SIFs without retyping or copying.

A SIF that contains another SIF (known as a sub-SIF) is described as a "host SIF". The sub-SIF resides in the sensor or final element subsystem of the host SIF, and is treated as a group within the host subsystem.



To add a sub-SIF, go to the SIF that will become the sub-SIF, and select the "Use this SIF as a sub-SIF" command in the SIF menu. You are then prompted to navigate to the subsystem of the desired host SIF, which can be done by using the dropdown box "Go to SIF" and the tree view (left side of the screen). Then click the "Attach" button in the bottom section of the screen.

Schematic diagram of example SIF with sub-SIFs as modelled by SILability



After adding a sub-SIF, you will need to change the architecture of the host subsystem to match the new number of groups plus sub-SIFs in the subsystem. For example, if the subsystem now contains one group and one sub-SIF, assign 1oo2 or 2oo2 architecture to the subsystem.

The sub-SIF and its host SIF must use the same operating mode and AC model.

4.8 TREE VIEW

The hierarchy of the currently selected SIF is shown in a tree view on the left side of the screen. To make the view more compact, you can collapse legs, groups and subsystems by clicking the small triangle on the left of each item.

Click an item in the tree view to view and edit the data in that item. For example, to edit the sensor subsystem, click “Sensors”. The corresponding data will appear in the data entry panel (upper right section of the screen).

4.9 ADDING GROUPS, LEGS AND COMPONENTS

You can add new elements to the SIF hierarchy in several ways:

- Select the parent of the element you want to create (e.g. to create a new leg, select a group that will contain it) by clicking it in the tree view. Then select Element menu - “Add new element”. (The menu text shows which kind of item will be added.) The new item will be added immediately below the currently selected item.
- Click <Add element> in the tree view.
- Duplicate an existing item. In the tree view, select the item you want to duplicate. Then, select Element menu – “Duplicate element”. This will create a copy below the selected element.

- Copy an existing item. In the tree view, select the item you want to copy. Then, select Element menu – “Copy element to another SIF”. The result panel (bottom right of the screen) now shows a “Copy here” button. Navigate to the destination for the new copy, using the “Go to SIF” choice box at the top of the screen and the tree view. Then click “Copy here”.

After adding or copying a group or leg, you will need to adjust the architecture of the parent to match the new number of items it contains. For example, change its architecture from 1oo1 to 1oo2 or 2oo2.

If you decide you don’t want the new element, use the Undo command in the Tools menu.

4.10 DELETING GROUPS, LEGS AND COMPONENTS

Select the element you want to delete by clicking it in the tree view. Select Element menu – “Delete element”. (The menu text shows which kind of element will be deleted.) If you delete a group or leg, you will need to adjust the architecture of its parent to match the number of elements remaining. For example, change its architecture from 1oo2 to 1oo1.

If you delete an item by mistake, use the Undo command in the Tools menu.

4.11 THE RESULT PANEL

The bottom right area of the screen is called the “result panel”, and shows the calculation results for the SIF element currently selected. To see the results for the overall SIF, select the SIF by clicking the topmost item in the tree view.

You can keep the current element in the result panel by clicking the “Stay” button at the top right of the result panel. This allows you to see the effect of data changes in other parts of the SIF. To release the “stay” function, click the “Release” button.

4.12 THE SIF LIST

You can view all the SIFs in the project in a summary table by selecting Tools menu – Show SIF list. The table shows:

- which SIFs are used as sub-SIFs (the symbol ◉➤ appears in the “Sub-SIF” column)
- which SIFs contain sub-SIFs (the symbol ➤◉ appears in the “Sub-SIF” column)
- The SIL target and SIL achieved for each SIF. If the SIL achieved cannot be determined, this column shows <ND>.

You can perform the following tasks in the SIF list:

- Filter the list to show a partial list of the SIFs in the project, using the “Filter:” choice box. To filter on text in the SIF’s Name or Tag fields, select “Name contains:” or “Tag contains:” in the Filter choice, and type the filter text in the text box to the right.
- Double click on any SIF in the table to exit the SIF list and jump to that SIF.
- To move a SIF to a different position in the list, click on the SIF’s number (leftmost column) and drag it to the new position in the table. You can undo this using the Undo command in the Tools menu.

When one or more SIFs are selected, you can perform the following actions by selecting them in the “Action:” choice box:

- deselect the SIFs
- invert the selection
- move the selected SIFs to the top or end of the SIF list
- swap the selected SIFs (if you selected exactly 2 SIFs)
- delete the selected SIFs.

To select which SIFs to act upon, click or Shift + Click or Ctrl + click (on Mac: Command + click) on the required SIFs. The move, swap and delete actions can be undone and redone using the Undo and Redo commands in the Tools menu.

SILability can also produce a SIF list for export into a report, using the “Export SIF list” command in the File menu. See section 9.2 for details.

5. CHAPTER 5 - DATA HANDLING FEATURES

5.1 INTRODUCTION

SILability provides a number of features to assist with entering the data needed to perform SIL verification calculations. These features are described in this chapter.

5.2 ENTERING DATA INTO TEXT FIELDS

Text fields designed to contain small amounts of information, such as “SIF Tag”, can contain only a single line of text. You can navigate to and from these fields by clicking in the field, or using the Tab or Shift + Tab keys. Your entry is stored as soon as you exit the field.

Other text fields such as “SIF Description” are multi-line fields. These fields will stretch to contain as much data as you wish to provide. You can start a new line using the Enter key, and insert a tab using the Tab key. To move to another field, click in the next field.

5.3 ENTERING DATA INTO NUMERICAL FIELDS

Numerical fields, such as lambda values (failure rates) in components, can contain only numbers. Initially, the value is undefined and the field shows “Not set”. To set the value, just enter the field and type a number. There’s no need to delete the words “Not set”; they will disappear automatically when you exit the field. Once the value is set, you cannot unset it (except by using Undo); even if you type “Not set”, it will be treated as an unrecognised value and the field will be left unchanged.

You can enter any value you wish in a numerical field, even if the value is invalid (such as a negative value for lambda). You can use scientific notation, for example by entering 1e-2 or 1E-2 for 0.01.

The value is stored and displayed with a predefined level of precision—eg 3 significant figures for lambda values—irrespective of how much precision you enter. For instance, if you enter 1.23456 in a lambda field, it will change to 1.23 when you exit the field. This is to prevent the user from claiming a misleading degree of precision for input data.

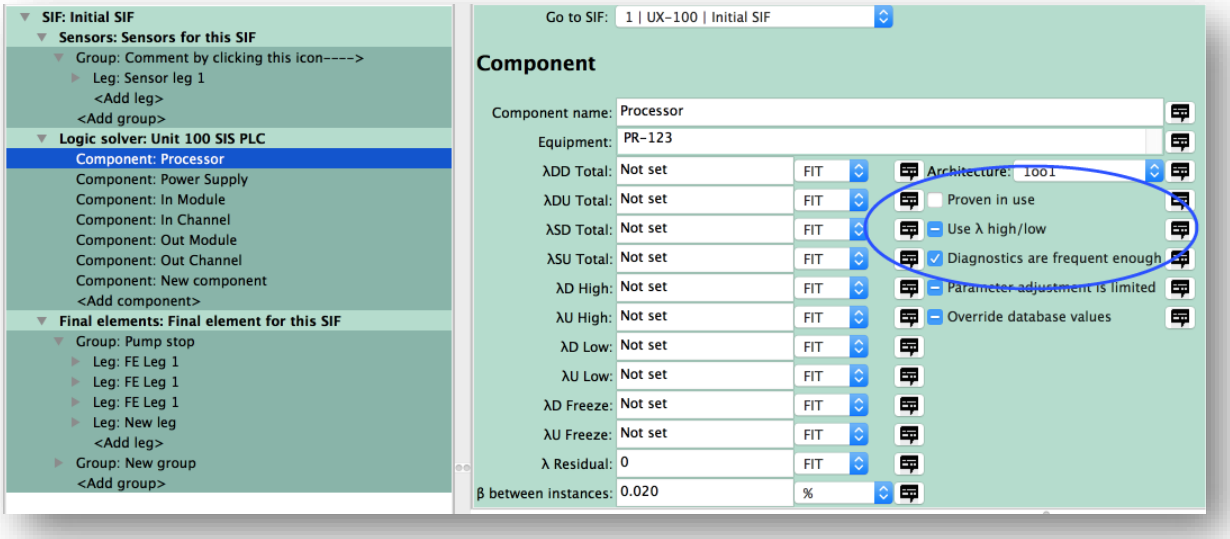
5.4 SETTING THE UNITS OF NUMERICAL FIELDS

Most numerical fields have a “unit” choice box to their right. Select the unit matching the value you entered. For example, when entering a β (common cause factor) value, you can enter an absolute value (say 0.1) with a unit of “no unit”, or a percentage (say 10%) with a unit of “%”. If you change the unit after entering the value, the value will NOT be automatically converted to match the new unit. This is to reduce the risk of values unintentionally changing without the user being aware of it.

For each component, the lambda (failure rate) value units are locked together. If you change one, they will change for all lambda values in that component. This is because lambda values are normally all provided with the same unit, so it is very unlikely that you would intentionally set one lambda unit differently from all the others within a single component.

5.5 SETTING CHECKBOX FIELDS

Checkbox fields, such as “Proven in use” in components, are initially undefined (neither checked nor unchecked). This is displayed as ▣ in Windows and a blue box with white horizontal line in MacOS. When you click the checkbox, it will change to checked status. You cannot “undefine” it again (except by using Undo).



5.6 UNDO AND REDO

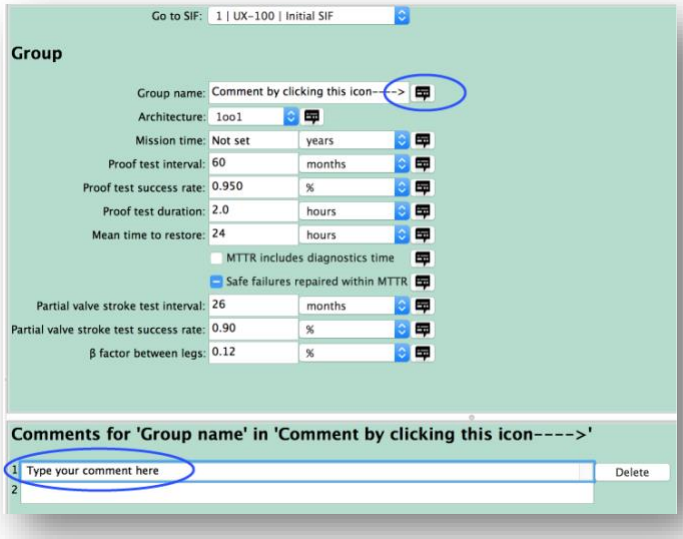
SILability provides full Undo and Redo functionality. Any change to the data can be reverted using the Undo command in the Tools menu. Unlimited undo is provided all the way back to the last time the project file was opened (or created, if it's a new project).

The Redo command in the Tools menu allows you to "undo an undo", if you decided you wanted to keep the original change after undoing it. If you undo a series of changes, you can redo each one in turn until they have all been redone.

When you undo or redo a change to a parameter that affects the SIL verification result, it will be immediately recalculated and displayed.

5.7 APPLYING COMMENTS TO DATA

SILability provides a powerful commenting function that allows you to attach comments to any item of data. For example, you can document the source of each item of data, record any assumptions made, or explain any differences between similar items of data. The comment system could also be used to track changes, or to record the name of the person entering each item of data.



To enter a comment, click the comment button to the right of the data field. The comment display now appears in the result panel at the bottom right of the screen. Click and type in the empty comment field. When you've finished, click the “Done” button or simply click anywhere else in the SILability window; your entry will be stored automatically.

If you want to enter another comment for the same data item, simply click on the comment button again. Another comment field will appear. A “Delete” button is provided to allow you to delete each comment.

All comments (with certain exceptions, see FAQs in Appendix H) will be shown in the spreadsheet report (see chapter 8).

Comment entry, editing and deletion can be undone and redone using the Undo and Redo commands in the Tools menu.

5.8 THE SANDBOX

If you want to make trial changes to the data in your SIFs to investigate the effect on the overall result, you can use the sandbox function. While the sandbox is in use, all data changes (to the project and to any open user database) made are temporary, and “save on the fly” is paused. When you have finished, you can discard all the changes you have made; this restores the data set to its state when you started the sandbox. Alternatively, you can opt to keep the changes.

To start the sandbox, use Tools menu – “Start sandbox” command. The result panel (lower right area of the screen) turns sand-coloured.

To end the sandbox, use Tools menu – “End sandbox, keep changes” or “End sandbox, discard changes” commands.

The Undo and Redo commands work as normal during the sandbox session. You can undo and redo any changes made, even after ending the sandbox.

If you close the project while the sandbox is in use, SILability asks if you want to keep or discard the changes made (to the project and any open database) during the sandbox session.

6. CHAPTER 6 - INPUT DATA

6.1 INTRODUCTION

In this chapter, we describe all the data needed for the SIL verification calculation. Each level of the SIF hierarchy—the SIF itself, subsystems, groups, sub-SIFs, legs and components—has a number of fields for data input. You can provide the data by:

- typing in the input data fields in the upper right panel of the screen,
- adding components from a SILability database (see chapter 7), or
- preparing data in an XML file and opening it as a project file.

The tables below show all the parameters for each level of hierarchy, along with an explanation of how SILability uses the data.

6.2 SIF-LEVEL DATA FIELDS

Parameter name	Purpose	Required for calculation?
SIF name	A brief, human-readable name such as “High level trip in steam drum V-100”	No, only for reporting purposes
SIL tag	The tag number of the SIF itself, as shown in the P&ID or C&ED. Typically begins with ‘UX’.	No, only for reporting purposes
SIF description	A description of the function of the SIF, in terms of sensors and final elements. Example: “On PAHH-123 (2oo3): close XV-200 and close XV-201 (2oo2).”	No, only for reporting purposes
SIF reference	The reference number and revision number of the document that defines the SIF, such as an SRS, C&ED or ESD narrative.	No. Important for traceability, FSA and configuration management purposes
Hazard	The process hazard that the SIF is designed to detect. Example: “Overfilling of steam drum V-100”. This can usually be copied from a HAZOP report, SIL determination report, or SRS.	No, only for reporting purposes
Consequence	The impact on risk receptors in the event of failure of the SIF under consideration and all other relevant layers of protection. Example: “Water carryover to steam header leading to hammering and damage. Potential operator injury, downtime for repair.”	No, only for reporting purposes
Process area	The unit or area containing the main equipment associated with the hazard.	No, only for reporting purposes
Operating mode	The operating mode of the SIF per IEC 61508 and IEC 61511: low demand mode, high demand mode or continuous mode. If the project uses IEC 61511:2003, select “Low demand mode” for	Yes

	demand mode. If the project uses IEC 62061, select high demand mode or continuous mode, depending on the relative frequency of diagnostic tests and demand on the SIF.	
AC model	The hardware fault tolerance (HFT) of the SIF can be evaluated against any of the following architectural constraint requirement models: IEC 61508:2010 Route 1H, IEC 61508:2010 Route 2H, IEC 61511:2003, or IEC 61511:2016. Alternatively, the requirement can be waived. This field allows you to select which model to apply.	Yes
SIL target	The SIL target selected for the SIF during a previous SIL determination.	No effect on the calculation. The report will indicate whether the SIL achieved by the SIF meets the target.
RRF or PFH target	The SIF's risk reduction target defined in a previous SIL determination. In low demand mode, this field contains the RRF target. In other operating modes, this field contains the PFH target.	No effect on the PF calculation. The report will indicate whether the PF achieved by the SIF meets the target. This parameter is used in the AC model IEC61508:2010 Route 2H.

6.3 SUBSYSTEM DATA FIELDS

Parameter name	Purpose	Required for calculation?
Subsystem name	A descriptive name for the subsystem in the SIF. Example: "Valves to prevent overfilling of V-100"	No, only for reporting purposes
Architecture	Sets the voting among sensor groups. Permitted architectures are 1oo1, 1oo2, 2oo2, 1oo3, 2oo3, 3oo3, 2oo4, and 4oo4. The *N* value in *M*oo*N* must match the number of groups defined in the sensor subsystem. If you select 2oo3 or 2oo4, all groups in the sensor subsystem must be identical; this is a restriction in SILability.	Yes
β (Beta)	Defines the common cause factor between groups. This is the fraction of failures (both dangerous and safe) that can be attributed to a common cause, such as an external stressor or a design flaw. Refer to IEC 61508:2010 part 6 for guidance in selecting a value. The value cannot be greater than the β values for any of the groups in the SIF. The rationale is that stressors leading to common cause failure *between* groups should be fewer than stressors causing failure *within* groups, because groups are generally more diverse than legs.	Yes, if architecture is redundant (*M*oo*N* architectures with *M*<*N*, e.g. 1oo2). For *M*oo*N* architectures with *M*>1, β is used to calculate MTTFS.
Reduce HFT requirement	IEC 61508:2010-2 architectural constraint Route 2H clause 7.4.4.3.2 provides an alternative rule set for cases where	Used for AC models IEC 61508:2010 Route 2H and IEC 61511:2016.

Parameter name	Purpose	Required for calculation?
	additional redundancy can introduce hazards. IEC 61511:2016-1 clause 11.4.6 has a similar provision. This checkbox allows you to select the alternative rule set, which generally leads to a lower HFT requirement. If you select IEC 61508:2010 Route 2H or IEC 61511:2016, this checkbox must be set (either checked or unchecked; its default state is undetermined). If you select any other AC model, the checkbox is disregarded.	

6.4 ADDITIONAL DATA FIELDS FOR LOGIC SOLVER SUBSYSTEM

Parameter name	Purpose	Required for calculation?
Equipment	The brand and model of hardware used to implement the logic solver.	No, only for reporting purposes
Name in database	(For future version) If the logic solver was selected from a database, this shows the name of the logic solver in the database.	No, only for reporting purposes
Category	(For future version) The category of the equipment, such as safety PLC, relay-based logic solver.	No, only for filtering in database
Trip action	Whether the logic solver implements the SIF by energising or de-energising. This option allows SILability to select the appropriate lambda values, as explained in section 6.7.	Refer to section 6.7
Mission time	The planned lifetime of the equipment used to implement the logic solver. The calculation assumes the equipment will be replaced with new equipment, or refurbished to as-new condition, at the end of the mission time. Mission time is counted from the date the equipment leaves the factory, not the date it is brought into service, as some components may start to deteriorate immediately.	Yes
Proof test success rate	The fraction of proof tests that are expected to be performed successfully. This provides a way to allow for the fact that proof tests may occasionally be delayed, performed incorrectly, or give false positive results. For PLCs, xSeriCon's suggested value is 0.95~0.98.	Yes

Parameter name	Purpose	Required for calculation?
β (Beta)	Same as β for other subsystems, except that it refers to common cause failures between components, rather than groups.	Yes, if architecture is redundant (*MooN* architectures with *M*<*N*, e.g. 1oo2). For *MooN* architectures with *M*>1, β is used to calculate MTTFS.
Mean time to restore (MTTR)	The mean time required to repair a fault in the subsystem, and test and recommission the subsystem, starting from the time the fault is discovered.	Yes
Proof test interval (PTI)	The planned interval between manual proof tests of the subsystem. If no proof testing is planned, set this value to longer than the Mission Time for the subsystem. (SILability will not allow you to simulate no-proof-testing by setting Proof Test Coverage to zero, because Proof Test Coverage must be greater than diagnostic coverage.)	Yes
Proof test duration	The time period per test that the subsystem is unavailable due to undergoing manual proof testing.	Yes
MTTR includes diagnostics time	Whether the MTTR includes a time allowance for the interval between automatic diagnostic tests of the subsystem. This is required when AC model IEC 61508:2010 Route 1H is selected, and can normally be set to Yes (checked).	Used for AC model IEC 61508:2010 Route 1H in Low Demand mode.
Detect safe failures in sensors	Whether the logic solver can raise an alarm (instead of a trip) in the event of a detected safe failure of a component in the sensor subsystem. This determines whether detectable safe failures in the sensor subsystem contribute to the MTTFS.	Used to calculate MTTFS.
Safe failures repaired within MTTR	If an undetected safe failure occurs in a logic solver with *MooN* architecture where *M*>1, it will not cause a spurious trip. However, it may be repaired if it is revealed (e.g. by a	Used to calculate MTTFS.

Parameter name	Purpose	Required for calculation?
	discrepancy alarm between channels of the logic solver). Set this checkbox to checked if undetected safe failures are likely to be revealed and repaired within the MTTR. In practice, it is unlikely that undetected safe failures in *MooN* architecture would be discovered until the next proof test, and the checkbox should normally be unchecked.	
Programmable	Whether the logic solver is programmable.	Used for AC model IEC 61511:2003, and for MTTFS calculations.

6.5 GROUP LEVEL PARAMETERS

Parameter name	Purpose	Required for calculation?
Group name	A descriptive name for the group. Example: “Level sensors in V-100”	No, only for reporting purposes
Architecture	Sets the voting among legs in the group. Permitted architectures are 1oo1, 1oo2, 2oo2, 1oo3, 2oo3, 3oo3, 2oo4, and 4oo4. The *N* value in *M*oo*N* must match the number of legs defined in the group. If you select 2oo3 or 2oo4, all legs in the group must be identical; this is a restriction in SILability.	Yes
β (Beta)	Defines the common cause factor between legs. This is the fraction of failures (both dangerous and safe) that can be attributed to a common cause, such as an external stressor or a design flaw. Refer to IEC 61508:2010 part 6 for guidance in selecting a value. The value cannot be greater than the β values for all the components in the group. The rationale is that stressors leading to common cause failure between legs should be fewer than stressors causing failure between identical components, and redundant components are assumed to be identical.	Yes, if architecture is redundant (*M*oo*N* architectures with $M<N$, e.g. 1oo2). For *M*oo*N* architectures with $M>1$, β is used to calculate MTTFS.
Mission time	The planned lifetime of the equipment used to implement the group. The calculation assumes the equipment will be replaced with new equipment, or refurbished to as-new condition, at the end of the mission time. Mission time is counted from the date	Yes

Parameter name	Purpose	Required for calculation?
	the equipment leaves the factory, not the date it is brought into service, as some components may start to deteriorate immediately.	
Proof test success rate	The fraction of proof tests that are expected to be performed successfully. This provides a way to allow for the fact that proof tests may occasionally be delayed, performed incorrectly, or give false positive results.	Yes
Partial valve stroke test success rate	The fraction of PVSTs that are expected to be performed successfully. This provides a way to allow for the fact that PVSTs may occasionally be delayed, performed incorrectly, or give false positive results. This parameter applies only to groups in the final element subsystem.	Yes
MTTR	Mean time to restore: this is the mean time required to repair a fault in the group, and test and recommission the group, starting from the time the fault is discovered. See FAQ's in Appendix H for further details.	Yes
Proof test interval	The planned interval between manual proof tests of the group. If no proof testing is planned, set this value > Mission time for the group. It is assumed that the whole group is tested at one time. However, if the group contains redundant legs, it is assumed that not all the legs are taken offline for testing simultaneously, so that protection is maintained during the proof test.	Yes
Partial valve stroke test interval	The planned time interval between PVST's. If no PVST is planned, uncheck the "Use PVST" checkbox in all of the legs of this group.	Yes, if any leg in this group has "Use PVST" switched on.

Parameter name	Purpose	Required for calculation?
	This parameter applies only to groups in the final element subsystem.	
Proof test duration	The time interval per test that the group is unavailable due to undergoing manual proof testing. If no proof testing is planned, set this value to zero. See Proof test interval (above) for more details.	Yes
MTTR includes diagnostics time	Whether the MTTR includes a time allowance for the interval between automatic diagnostic tests of the group. This is required when AC model IEC 61508:2010 Route 1H is selected, and can normally be set to Yes (checked).	Used for AC model IEC 61508:2010 Route 1H in Low Demand mode.
Safe failures repaired within MTTR	If a safe failure occurs in an element with *M*oo*N* architecture where *M*>1, it will not cause a spurious trip. However, it may be repaired if it is revealed (e.g. by a bad PV alarm or discrepancy alarm). Set this checkbox to checked if undetected safe failures are likely to be revealed and repaired within the MTTR. In practice, this is plausible for elements in the sensor subsystem. However, in a final element subsystem with *N*oo*N* architecture, it is possible that undetected safe failures may not be discovered until the next proof test, in which case the checkbox should be unchecked.	Used for MTTFS calculation.

6.6 LEG LEVEL PARAMETERS

Parameter name	Purpose	Required for calculation?
Leg name	A descriptive name for the leg. Example: “Level sensors leg 1”. This could include the device tag number.	No, only for reporting purposes
Use PVST	Whether you intend to implement partial valve stroke testing of the leg. Applies only to legs in the final element subsystem, and only to legs containing process valves.	Yes

6.7 COMPONENT LEVEL PARAMETERS

Parameter name	Purpose	Required for calculation?
Component name	The tag number or other unique identifier for the component	No, only for reporting purposes
Equipment	The physical equipment (brand name and model number) used to implement the component. You are advised not to put the tag	No, only for reporting purposes and for storage in the database

Parameter name	Purpose	Required for calculation?
	number here, as it could be stored in the database, causing confusion when a component is retrieved from the database.	
Data source	A reference to the source of information used, especially the failure rates.	No, only for reporting purposes and for storage in the database
Name in database	If the component was selected from a database, this shows the name of the device in the database.	No, only for identification in the database
Category	The category of the equipment, such as ball valve, actuator, pressure sensor, IS barrier.	No, only for filtering in database
Database comment	A comment stored in the database record for this component. You can use this for any purpose you wish; for example, the name of the person who added the component to the database, or the name of the project from which it was added.	No, only for storage in the database
Architecture	Whether the component is provided with internal redundancy. For example, if two identical limit switches or two solenoid valves are provided, and configured so that successful operation of either of them can achieve the design intent of the component, you can set the architecture to 1oo2. The options are 1oo1 and 1oo2.	Yes
Hardware type	Type A or Type B as defined in IEC 61508:2010. In essence, Type A are simple non-programmable devices, such as limit switches, valves and thermocouples. Type B are programmable devices.	Yes, if IEC 61508:2010 Route 1H or Route 2H are selected as AC model
λDD Total	The component's total rate of dangerous random failures that are detectable by diagnostics. This is typically the λ_{DD} value provided in the device's safety manual or SIL capability certificate.	Yes, if trip direction/action is not defined, or if 'Use λ high/low' is unchecked.

Parameter name	Purpose	Required for calculation?
	A dangerous failure is defined as any failure that can prevent the SIF from putting the process (or EUC) into the defined safe state, in the absence of redundancy. If you define the trip direction (high/low) or action (e.g. energise/de-energise, open/close) and you have separate λ_D values for these, you can use these λ_D values instead of λDD Total by setting 'Use λ high/low' to checked.	
λDU Total	The component's total rate of dangerous random failures that are not detectable by diagnostics. This is typically the λ_{DU} value provided by the manufacturer. For further details see λDD Total.	Yes, if trip direction/action is not defined, or if 'Use λ high/low' is unchecked.
λSD Total	The component's total rate of safe random failures that are detectable by diagnostics. This is typically the λ_{SD} value provided by the manufacturer. For further details see λDD Total. A safe failure is defined as any failure that can cause a spurious trip of the SIF, unless the device is in *MooN* architecture where *M*>1 (in which case, *M* such failures can cause a spurious trip), or the trip is suppressed by diagnostics.	Yes, if trip direction/action is not defined, or if 'Use λ high/low' is unchecked.
λSU Total	The component's total rate of safe random failures that are not detectable by diagnostics. This is typically the λ_{SU} value provided by the manufacturer. For further details see λDD Total.	Yes, if trip direction/action is not defined, or if 'Use λ high/low' is unchecked.
λ Residual	The component's total rate of random failures that are neither safe nor dangerous. A typical example is failure of a display or a diagnostic.	No. This is provided for possible future use.

Parameter name	Purpose	Required for calculation?
λD High	The component's total rate of random failures, detectable by diagnostics, that cause it spuriously to give a high signal, energise, open or remain in energised/open state. For further details see λDD Total.	Yes, if trip direction/action is defined and 'Use λ high/low' is checked.
λD Freeze	The component's total rate of random failures detectable by diagnostics, that cause its analogue output to show an incorrect value that could be higher or lower than the true value. For further details see λDDTotal.	Yes, if trip direction/action is defined and 'Use λ high/low' is checked.
λD Low	The component's total rate of random failures, detectable by diagnostics, that cause it spuriously to give a low signal, de-energise, close or remain in de-energised/closed state. For further details see λDD Total.	Yes, if trip direction/action is defined and 'Use λ high/low' is checked.
λU High	The component's total rate of random failures, not detectable by diagnostics, that cause it spuriously to give a high signal, energise, open or remain in energised/open state. For further details see λDD Total.	Yes, if trip direction/action is defined and 'Use λ high/low' is checked.
λU Freeze	The component's total rate of random failures not detectable by diagnostics, that cause its analogue output to show an incorrect value that could be higher or lower than the true value. For further details see λDD Total.	Yes, if trip direction/action is defined and 'Use λ high/low' is checked.
λU Low	The component's total rate of random failures, not detectable by diagnostics, that cause it spuriously to give a low signal, de-energise, close or remain in de-energised/closed state. For further details see λDD Total.	Yes, if trip direction/action is defined and 'Use λ high/low' is checked.
Trip direction *or* Trip action	For components in the sensor subsystem, this defines the sense of the trip function. If the function is designed to trip when a	Yes

Parameter name	Purpose	Required for calculation?
	process variable is greater than a threshold value, or when a logic 1 signal is generated (eg from a limit switch), the trip direction is “high”. If the function should trip when the process variable is below a threshold, or a logic 0 signal is generated, the trip direction is “low”. For components in the logic solver and final element subsystems, the options are “open to trip” and “close to trip” (for valves), or “energise to trip” and “de-energise to trip” (for other devices). This value is used to determine which failure rates (λ values) are used in the PFDavg/PFH and MTTFS calculations. For example, if the trip direction is set to “high”, λHigh is taken as a safe failure rate, while λFreeze and λLow are taken as dangerous failure rates. If you don’t have sufficiently detailed λ data, you can set the Trip direction or Trip action to “Undefined”, or set ‘Use λ high/low’ to unchecked. SILability will then use the λTotal values to determine failure rates, and λHigh, Low and Freeze values will be ignored.	
β between instances	Defines the common cause factor between instances of identical, redundant components. This is the fraction of failures (all types) that can be attributed to a common cause, such as an external stressor or a design flaw. Refer to IEC 61508:2010 part 6 for guidance in selecting a value.	Yes, if architecture is redundant (1oo2).

Parameter name	Purpose	Required for calculation?
Proof test coverage	The fraction of all failures (dangerous and safe; detectable and undetectable) that are covered by the device's proof test method. (In a future version, you will be able to enter 3 separate values: PTC High covers failures included in λDHigh and λUHigh; similarly for PTC Low and PTC Freeze.) As some devices have several possible test methods, make sure to use the proof test coverage quoted for the method you intend to apply. In low demand mode, the PFDavg can be quite sensitive to this parameter, so it is important not to overestimate it.	Yes
Partial valve stroke test coverage	The fraction of failures covered by the device's partial valve stroke test method. Applies to components in the final element subsystem only. If you are not using PVST, set this parameter to 0. If you leave the value unset, it is assumed to be 0. (In a future version, you will be able to enter separate values for PVST coverage open and close, for PVST's that open and close the valve respectively.)	Yes
Proven in use	Whether the device has undergone a 'proven in use' or 'prior use' assessment. You can also set this to checked if the device is SIL certified by a method that includes prior use assessment.	Used in IEC 61511:2003 AC model.
Override database values	If this is set to checked, you can manually override any λ values (and other parameters) retrieved from a SILability database, by typing new values in the respective fields. Otherwise, if the device was selected from a database, these fields are locked for editing.	No

Parameter name	Purpose	Required for calculation?
	If checked, this field also suppresses automatic updating of the device's λ values if the corresponding database item is changed. See chapter 7 for details on using a database.	
Use λ high/low	If this is set to checked, and if the Trip action/direction field is set to a value other than “Undefined”, the calculation will use λD High, λD Freeze, λD Low, λU High, λU Freeze, and λU Low values for this device. Otherwise, the calculation will use λDD Total, λDU Total, λSD Total, and λSU Total.	Yes
Diagnostics are frequent enough	Set this to checked if the sum of the device’s diagnostic test interval and time required for the SIF to perform the safety action or to maintain a safe state is less than the process safety time.	Used in IEC 61508:2010 AC Route 1H for high demand and continuous modes. For other AC models and low demand mode, the parameter is ignored.
Parameter adjustment is protected	Set this to checked if the device allows only limited adjustment of parameters (not full programming) and such adjustment is protected (e.g. by password). For devices with no parameter adjustment (eg switches, RTDs, valves), this can always be set to checked.	Used in IEC 61511:2003 AC model.

6.8 ENGINEERING UNITS

All numerical parameters provide a field for you to set the engineering unit of your choice. For example, mission time can be given in hours, days, weeks, months or years. You can freely mix the units within the project; for example, proof test interval can be given in months and partial valve stroke test interval in weeks, if you wish. Prior to calculation, SILability converts all values to a fixed set of units internally.

However, if you change the unit of any λ value in a component, the units of all other λ values in the same component change automatically. *Linked values in other components will also change units automatically* (see chapter 5 for an explanation of linking). The numerical values are not changed. This is to reduce the risk of accidentally setting the unit of one λ different from the others; most likely, the source of the λ data will use the same unit for all λ values.

All times refer to calendar time, rather than time in service, because it is assumed that random hardware failures can occur even when equipment is not in service.

7. CHAPTER 7 – FAILURE RATE DATABASES

SILability is provided with an extensive database containing over 3,000 sets of failure rates for hardware components. This database is opened automatically when SILability starts. All data in this database is from reliable, publicly available sources, and labelled with the source of data for traceability.

In addition, the user can build up his/her own custom database.

These two databases can be accessed simultaneously within SILability, and are known as the "xSeriCon database" and the "user database".

7.1 FETCHING A COMPONENT FROM THE DATABASES

Components from the databases can be inserted as new components into a SIF. First, navigate to the SIF that will receive the new component. Go to an existing leg or component, or to the logic solver subsystem.

Navigate here…	To insert the new component here…
A leg within the sensor or final element subsystem	In the leg, before any existing components
An existing component	Immediately after the selected component
The logic solver subsystem	In the subsystem, before any existing components

In the Database menu, select "Get component from database".

You can now select a component from the databases, filtering on category if desired. Select the database name of the component in the "Component name" dropdown box. The parameter values stored in the databases are shown in the data entry panel (in the middle of the screen). Click Apply to transfer the parameter values to the component in the project. This will lock the component to the database, so that you can't make further changes until you set 'Override database values' to checked.

You can use Undo to remove the component from your SIF if needed.

The following parameters in the project component still need to be defined manually, as they are application-specific. See chapter 6 for an explanation of these parameters.

- Trip direction/action
- Architecture
- Diagnostics are frequent enough

7.2 LOCKED AND UNLOCKED COMPONENTS

When a component's parameters are fetched from a database, the project component is locked to the corresponding database component. This is to prevent inadvertent changes to the project data. The component's 'locked' status is shown by the words "(locked to database)" in the data entry panel.

If you try to change a database-related parameter in a locked component, the change is rejected and the 'Override database values' checkbox is highlighted as a reminder.

If you wish to change database-related parameters (as listed in section 7.2), you need to unlock the component by checking its Override database values checkbox. The component's status will be shown as "(from database, overridden)".

To lock the component again, uncheck Override database values. **This will revert all the component's database-related parameters to the values stored in the database.**

All these actions can be reversed with Undo if needed.

7.3 GETTING STARTED WITH USER DATABASES

SILability allows you to build up your own database of components for use in SIFs. All the parameters needed to perform SIL verification are stored in the database, along with identification fields and a comment field.

First, make a copy of the empty database file provided with your SILability package; give your copy a meaningful filename such as "MyDatabase.SILabilityDatabase".

The database is stored in an XML file in any location of your choice. The file can be shared between projects and among users. Please contact xSeriCon if you require details of the file structure. SILability database files have the file extension .SILabilityDatabase.

To start populating a database, you need to open a SILability project (see chapter 3).

Next, open your copy of the empty database using the Database menu – Open database command. In the dialogue that appears in the data entry panel, click Select and select your empty database file such as "MyDatabase.SILabilityDatabase".

If you want your work on the database to be saved on the fly, click "Save changes as I work".

Then click "Go" to open the database.

You can only open one user database at a time. If you need to fetch a component from another database, close the current database first (see section 7.6).

7.4 POPULATING YOUR DATABASE

To add a component to your database, first set up the component in a SIF in your project. Enter all the component parameters you want to store in the database. The following parameters will be stored in the database (see chapter 6 for details of what the parameters mean):

- Equipment
- Data source
- All λ values
- β between instances
- Proof test coverage
- Partial valve stroke test coverage (for final element components)
- Proven in use
- Use λ high/low
- Hardware type (A/B)
- Parameter adjustment is limited

Other component-related fields (such as architecture) are not stored because they are implementation-dependent – their value varies from one project to another.

In the Database menu, select Store component in database (or Update component in database, if the component already exists in the database; see below). The data entry panel changes to show the "Write component to database" view.

Enter the following information for storage in the database (see Chapter 6 for details of the meaning of each parameter):

- Component name in database: this is the name by which the component will be known in the database. It can be different from the component name in the project. This parameter cannot be left blank.
- Equipment: the brand name and exact model of hardware. This is the field used to match components in the project to components in the database. It cannot be left blank. (Any change you make in this field will be copied back to the component in the project when you click Go.)
  If the equipment's failure rates vary depending on the usage (such as trip action/direction, severe service and tight shutoff), specify the usage in the equipment name. For example, a ball valve's λ values for an open-to-trip application are different from close-to-trip; you should specify, for example, "ValveCo BV-100 open-to-trip".
- Category in database: a list of predefined categories is provided; select one of these, or type your own category. The category is useful for filtering purposes.
- Data source reference
- Comment in database: a comment field you can use for any purpose, such as the name of the person entering the data and the date. It is purely informational and does not appear in the SIL verification report.

If the database already contains a component with the same equipment but different parameters, the message “Component already exists in database” appears. When you click Go, the existing database component will be overwritten. If this creates discrepancies with other components in the project, the discrepancies need to be resolved; see section 7.7 later in this chapter.

7.5 SAVING YOUR DATABASE

To save your database in a .SILabilityDatabase file, go to the Database menu and select Save database.

When opening the database, if you click the ‘Save changes as I work’ checkbox, all subsequent database changes are saved immediately, and there is no need for you to save the database manually. You are highly recommended to use this option if you are building up the database, as it will avoid losing any work if there is a computer crash. However, if you are using an existing database and don’t want to save inadvertent changes, uncheck ‘Save changes as I work’.

7.6 CLOSING YOUR DATABASE

To close your user database, go to the Database menu and select Close database. If there are any unsaved changes, you can save them by clicking the Save button that appears. If there are no unsaved changes, the user database will close immediately. However, the xSeriCon database will remain open and available for use.

SILability may not allow you to close the database while the sandbox is in use (see section 5.8), because of potential data conflicts if you undo the database closing operation.

7.7 CHECKING WHICH DATABASE IS OPEN

The File menu - Project information command will show the filename of the currently open user database, if any. It also shows the version number of the xSeriCon database, and whether the user database is currently saving on the fly.

7.8 DISCREPANCY HANDLING

When you open a user database, or write a component into the database, SILability checks for any discrepancies between the database and the project. For example, if both the database and the project contain a component whose equipment is ‘Brand X pressure transmitter PT-100 MkII’ and they have different λ values, this counts as a discrepancy.

Discrepancies must be resolved before you can proceed with the project. Alternatively, if you were opening a database, you can abandon the database opening by clicking 'Cancel database opening' in the result panel at the bottom right of the screen.

Discrepancies are shown in the result panel. For each discrepancy, you can resolve it by selecting an action from the <Choose action> dropdown box:

- **Ignore database:** this will keep the values in the database and project unchanged, and set the 'Override database values' checkbox of the affected component to Yes.
- **Use database value:** this will transfer the parameter value from the database to the project component. Use this option if you trust the database.
- **Make component standalone:** this breaks the link between the database component and the project component. In the project component, the 'Name in database' parameter is cleared.
- **Overwrite database value:** this transfers the parameter value from the project component to the database. If the same component is used elsewhere in the project or in other projects, this may cause new discrepancies to appear. This option is available only if the component is in the user database; the xSeriCon database cannot be modified.

The 'Show in tree' button next to each discrepancy navigates the tree (left side of screen) to the affected component in the project, allowing you to see the affected SIF.

All actions taken to resolve discrepancies will be reverted if you click 'Cancel database opening.' Also, you can revert all the actions individually, as well as re-closing the database, using the Undo function.

7.9 PLCs IN THE DATABASE

This will be implemented in a future version of SILability.

8. CHAPTER 8 - THE SIL VERIFICATION CALCULATION

8.1 INTRODUCTION

This chapter provides a brief description of the calculations performed by SILability, and explains the meaning of the calculation results. If the input data is incomplete or out of range, SILability will raise a problem message and will not execute the affected part of the

calculation. Problem messages you may encounter, their meanings, and how to resolve them, are also covered in this chapter.

As explained in chapter 1, use of SILability assumes knowledge of functional safety engineering and the applicable standards, in particular IEC 61508 and IEC 61511. If any concepts or terms used in this chapter are unfamiliar to you, please refer to the standards or other resources or contact xSeriCon for further assistance and training. Some suggested further reading is listed in Appendix G of this user guide.

8.2 METHODOLOGY

Failure measures are calculated in terms of Probability of Failure on Demand of the SIF, averaged over the lifetime of the SIF (PFDavg), for SIFs in low demand mode. The result is displayed as a Risk Reduction Factor (RRF), which is simply 1/PFDavg. If the SIF is in high demand or continuous mode, the failure measure is the SIF's Probability of Failure per Hour (PFH).

SILability uses a 'simplified equations' approach to calculating the failure measures. For low demand mode, the time-dependent probability of failure is calculated for each element of the SIF, with a sampling interval of 3 months (or the shortest test interval used, whichever is less), and compounded together prior to time-averaging. The calculation takes into account the unavailability of each device due to proof testing, restoration after discovery of a fault by proof testing, and restoration after discovery of a fault by diagnostics. A carefully selected set of assumptions is made, as listed in Appendix B.

The calculation generates the following outputs for each SIF, if sufficient data has been input:

8.2.1 SIF level

Parameter	Notes
SIL achieved	The maximum SIL achieved by the SIF as a whole, taking PFDavg/PFH and architectural constraints (AC) into account
PFDavg/PFH	The PFDavg or PFH (depending on the operating mode) achieved by the SIF as a whole
Failure rate target met	A boolean (yes or no) value indicating whether the SIF has met its PFDavg/PFH target
Risk Reduction Factor	The risk reduction factor (RRF) achieved by the SIF as a whole, if in low demand mode
Maximum SIL (failure rate)	The SIL achieved by the SIF as a whole, taking into account PFDavg/PFH only
Architectural constraints target met	A boolean (yes or no) value indicating whether the SIF has met its AC target, unless the AC model is set to 'Waived'
Maximum SIL (architectural constraints)	The SIL achieved by the SIF as a whole, taking into account architectural constraints only (unless the AC model is set to 'Waived')
PFDavg/PFH contributions from each subsystem	The percentage contribution to the overall PFDavg/PFH of the SIF from each subsystem
MTTFS (Spurious trip)	The mean time to fail spurious of the SIF as a whole. This is the predicted mean time between spurious trips caused by random hardware failures. If all applicable λ values are set to zero, this may be reported as ∞.

8.2.2 Subsystem level

Parameter	Notes
Maximum SIL achieved by subsystem	The notional SIL achieved by the subsystem, taking into account PFDavg/PFH and AC (unless the AC model is set to 'Waived')
Maximum SIL (failure rate)	The notional SIL achieved by the subsystem, taking into account PFDavg/PFH only
Maximum SIL (architectural constraints)	The notional SIL achieved by the subsystem, taking into account AC only (unless the AC model is set to 'Waived')
PFDavg/PFH	The PFDavg/PFH of the subsystem
HFT	The hardware fault tolerance of the subsystem (not shown for logic solver, as it can be deduced directly from the *MooN* architecture of the logic solver)
MTTFS	The mean time to fail spurious of the subsystem. This is the predicted mean time between spurious trips caused by random hardware failures. If all applicable λ values are set to zero, this may be reported as ∞.
PFDavg/PFH contributions per group/component	The percentage contribution to the overall PFDavg/PFH of the subsystem from each group or component in the subsystem. Not shown for certain architectures, if the group/component PFDavg/PFH is not directly related to the subsystem PFDavg/PFH.

8.2.3 Group level

Parameter	Notes
PFDavg/PFH	The PFDavg/PFH of the group
MTTFS	The mean time to fail spurious of the group. This is the predicted mean time between spurious trips caused by random hardware failures. If all applicable λ values are set to zero, this may be reported as ∞.
PFDavg/PFH contributions per leg	The percentage contribution to the overall PFDavg/PFH of the subsystem from each leg in the group. Not shown for certain architectures, if the leg's PFDavg/PFH is not directly related to the subsystem PFDavg/PFH.

8.2.4 Leg level

Parameter	Notes
PFDavg/PFH	The PFDavg/PFH of the leg
Safe failure fraction (SFF)	The ratio of the safe failure rate to the total failure rate of the leg, excluding residual ('no effect') failures. Calculated according to the method given in IEC 61508:2010-2.
PFDavg/PFH contributions per component	The percentage contribution to the overall PFDavg/PFH of the subsystem from each component (or redundant pair of components, if the component's architecture is set to 1oo2) in the leg.

8.2.5 Component level

Parameter	Notes
PFDavg/PFH	The PFDavg/PFH of the component (or redundant pair of components, if the component's architecture is set to 1oo2).
Safe failure fraction (SFF)	The ratio of the safe failure rate to the total failure rate of the component, excluding residual ('no effect') failures. Calculated according to the method given in IEC 61508:2010-2. The displayed SFF refers to a single instance of the component, even if the component's architecture is set to 1oo2. Thus, it can be used as a cross-check against any published SFF in the source you got the failure rate data from.

8.3 DATA CHECKS PERFORMED BEFORE CALCULATION

SILability calculates all the values listed above, every time you change the input data or settings for a SIF. This is done automatically and there is no need to request a recalculation. SILability never shows you invalid or out-of-date results.

Before calculating, SILability checks that the input data is complete and that all values are suitable. If any of the data is incomplete or unsuitable, part or all of the calculation may not be performed. When this occurs, SILability displays problem messages instead of the results. The problem messages you see are relevant to the portion of the SIF currently selected in the tree view (left side of the screen). For instance, if the sensor subsystem (or any group, leg or component in the sensor subsystem) is selected, you will see problem messages relating to the sensor subsystem only.

Next to each problem message is a 'Show me' button. This provides a shortcut to the data parameter that is causing the problem. For example, if a λ (failure rate) value is out of range, the 'Show me' button will make the data entry panel jump to the component containing that value, and the value will be highlighted.

The data checks are intended to ensure that the calculation can produce a meaningful result. They do not check that the data is consistent or within normal ranges. A separate check function is provided for this: see Chapter 9.

In the spreadsheet output (described in Chapter 8), any values that cannot be calculated are left blank.

Here is a complete list of the problem messages you may see, in alphabetical order.

8.3.1 Data problem messages

Message	Explanation	Possible solutions
All component AC types must be identical when using this group/subsystem architecture	For sensor and final element subsystems, if you select 2oo3 or 2oo4 group architecture, all legs and components within that architecture must be identical (this is a SILability restriction).	Adjust the parameters of components within the affected group or subsystem, to ensure they are identical.
Beta values must be identical across all components when using this group/subsystem architecture	Refer to “All component AC types must be identical when using this group/subsystem architecture”.	
‘Detect safe failures in sensors’ checkbox in logic solver subsystem is undefined	If the sensor subsystem has any safe detectable failure modes, SILability needs to know whether the logic solver can detect them.	• If you are not sure, set the logic solver’s “Detect safe failures” checkbox to unchecked. *or* • Make sure the logic solver's “Programmable” checkbox is set correctly. (If the logic solver is not programmable, SILability assumes it cannot detect safe failures, so there’s no need to define the “Detect safe failures” checkbox.)
‘Diagnostics are frequent enough’ checkbox must be defined when using Route 1H AC model in high demand or continuous mode	The IEC 61508:2010 AC Route 1H model needs to know whether any diagnostics for each component are frequent enough to find failures before a demand occurs (see Chapter 6 for details).	• If you don’t know the status of any component, set “Diagnostics are frequent enough” to unchecked. *or* • Check the SIF operating mode. *or* • Change to a different AC model (at SIF level).

Message	Explanation	Possible solutions
Each leg must have the same number of components when using this group/subsystem architecture	Refer to “All component AC types must be identical when using this group/subsystem architecture”.	
FVL/LVL legs must have DC ≥ 0.6 in IEC 61511:2016 AC model	In IEC 61511:2016 clause 11.4.8, there is a requirement that all FVL/LVL elements have diagnostic coverage ≥ 0.6. This message means that SILability has calculated the DC of a leg containing one or more FVL/LVL component(s) as < 0.6. (A component is considered to be FVL/LVL if it is type B and its ‘Parameter adjustment is limited’ checkbox is not checked.)	Check the component λ values in the leg are correct. Check that the component AC type (A or B) and ‘Parameter adjustment is limited’ checkbox are set correctly for all components in the leg. Use a different AC model (at SIF level).
Group β can't be greater than component β	In a group, SILability assumes that common cause failure between identical components is more likely than between legs, as components have relatively more common failure modes. Therefore, the β of the group should not be greater than the β of any component in the group.	Increase component β, or reduce group β. If you see this message unexpectedly, you may have set mismatched units for β (absolute value, such as 0.1, or percentage, such as 10%).
Group architecture must be selected	In each group that is included in the SIF, you must specify the architecture to be used in the calculation.	Set the architecture to any value *M*oo*N* where *N* is the number of legs in the group.
Group architectures must be identical when using this subsystem architecture	Refer to “All component AC types must be identical when using this group/subsystem architecture”.	

Message	Explanation	Possible solutions
Lambda values must be identical across all components when using this group/subsystem architecture	Refer to “All component AC types must be identical when using this group/subsystem architecture”.	
Logic solver β can’t be greater than component β	In a logic solver, SILability assumes that common cause failure between identical components is more likely than between “channels” (redundant processing pathways), as components have relatively more common failure modes. Therefore, the β of the subsystem logic solver should not be greater than the β of any component in the logic solver.	Increase component β, or reduce logic solver β. If you see this message unexpectedly, you may have set mismatched units for β (absolute value, such as 0.1, or percentage, such as 10%).
Logic solver architecture must be selected	In the logic solver, you must specify the architecture to be used in the calculation.	Set the architecture to any value *MooN* where *N* is the number of processing channels in the logic solver.
‘MTTR includes diagnostics’ must be defined when using IEC 61508:2010 Route 1H	The IEC 61508:2010 AC Route 1H model needs to know whether the MTTR value includes an allowance for the diagnostic test interval (see Chapter 6 for details).	• If you are not sure, set the group’s or logic solver’s “MTTR includes diagnostics” checkbox to unchecked. *or* • Change to a different AC model (at SIF level).
No components defined (need at least 1)	Each leg, and the logic solver subsystem, must contain at least one component.	In a leg: create and populate a component in the leg, or delete the leg. In the logic solver subsystem: create and populate a component, or switch off the logic solver by unchecking “Include logic solver in SIF”.

Message	Explanation	Possible solutions
No groups defined (need at least 1)	The sensor and final element subsystems must contain at least one group (or sub-SIF) each.	• Create a group in the subsystem and populate it with at least one leg and one component. *or* • Populate the subsystem with a sub-SIF. *or* • Switch off the subsystem by unchecking "Include this subsystem in SIF".
No legs defined (need at least 1)	Each group must contain at least one leg.	• Create a leg in the group and populate it with at least one component. *or* • You might have accidentally created an empty group. If so, delete the unwanted group and adjust the subsystem architecture to match the remaining groups.
Parameter Adjustment checkbox must be defined when using IEC 61511:2003 AC model	The IEC 61511:2003 AC model needs to know whether each component has restricted parameter adjustment capability.	• If you don't know the status of any component, set "Parameter adjustment is limited" to unchecked. *or* • Change to a different AC model (at SIF level).
PIU checkbox must be defined when using IEC 61511:2003 AC model	The IEC 61511:2003 AC model needs to know whether each component is "proven in use" (PIU).	• If you don't know the PIU status of any component, set it to unchecked. *or* • Change to a different AC model (at SIF level).
'Programmable' checkbox in logic solver subsystem is undefined	SILability needs to know whether the logic solver is programmable (i.e. whether it is a PLC, rather than a fixed logic solver such as a relay assembly).	Set the logic solver's "Programmable" checkbox. If the logic solver is a PLC, set it to checked.
Programmable logic solver must have DC ≥ 0.6 in Route 2H AC model	In IEC 61508 AC route 2H, there is a requirement that all type B elements have diagnostic coverage ≥ 0.6. This message means that SILability has calculated the DC of the logic solver as < 0.6.	• Check the component λ values in the logic solver are correct. *or* • Check that the "Programmable" checkbox of the logic solver is set correctly. *or* • Use a different AC model (at SIF level).

Message	Explanation	Possible solutions
Programmable logic solver must have DC ≥ 0.6 in IEC 61511:2016 AC model	Same as above, for IEC 61511:2016 AC model.	
'Programmable' must be defined	The logic solver needs to be defined as programmable or non-programmable. This information is used in the AC calculation.	Set the 'Programmable' checkbox to checked or unchecked. If you are using a PLC, you can set the checkbox to checked.
Proof test coverage can't be less than diagnostic coverage	Proof testing should be able to reveal a higher fraction of failure modes than diagnostics. (If this were not true, there would be no point in performing proof testing.) Therefore, each component's proof test coverage value should be higher than the diagnostic coverage calculated by SILability.	This problem could be caused by: • A very high diagnostic coverage due to incorrect λ values *or* • A low proof test coverage value *or* • A low proof test success rate value defined at group level
Proof test coverage values must be identical across all components when using this group/subsystem architecture	Refer to "All component AC types must be identical when using this group/subsystem architecture".	

Message	Explanation	Possible solutions
PVST coverage can't be less than diagnostic coverage	Partial valve stroke testing should be able to reveal a higher fraction of failure modes than automatic diagnostics. (If this were not true, there would be no point in performing stroke testing.) Therefore, each component's PVST coverage value should be higher than the diagnostic coverage calculated by SILability.	This problem could be caused by: • A very high diagnostic coverage due to incorrect λ values *or* • A low PVST coverage value *or* • A low PVST success rate value defined at group level *or* • 'Use PVST' checkbox in the leg set to Checked when PVST is not required
PVST coverage can't be less than proof test coverage	Proof testing should be able to reveal a higher fraction of failure modes than PVST. (If this were not true, there would be no point in performing proof testing.) Therefore, each component's proof test coverage value should be higher than its PVST coverage value.	This problem could be caused by: • A high PVST success rate value defined at group level *or* • A low proof test success rate value defined at group level If you intend to use only PVST and not proof test, you can set the group's proof test interval equal to the mission time and the proof test duration to 0. This will mean that proof testing has no effect in the calculation, so you can set proof test coverage to an arbitrary value of 100%.
PVST coverage value needed (or you can switch off PVST in the leg)	In a final element component, you must supply a PVST coverage value if you switched on PVST at leg level.	• Supply a PVST coverage value. *or* • Unselect "Use PVST" in the leg containing the component.

Message	Explanation	Possible solutions
PVST Coverage values must be identical across all components when using this group/subsystem architecture	Refer to "All component AC types must be identical when using this group/subsystem architecture".	
'Reduce HFT requirement' must be defined when using Route 2H or IEC 61511:2016	If you selected the IEC 61508:2010 Route 2H AC model, SILability needs to know whether you wish to invoke clause 7.4.4.3.2 of part 2 of the standard, to reduce the HFT requirement of each subsystem. Similarly for IEC 61511:2016 AC model.	• If you are not sure, set "Reduce HFT requirement" to unchecked. *or* • Change to a different AC model (at SIF level).
'Safe failures repaired within MTTR' checkbox must be defined when using this architecture	In a group or logic solver, if the *MooN* architecture has *M*>1, SILability needs to know if any undetected safe failures will be noticed and repaired within the MTTR.	If you are not sure, set the group's (or logic solver's) "Safe failures repaired within MTTR" checkbox to unchecked. (See Chapter 6 for more details.)
'Safe failures repaired within MTTR' must be identical when using this subsystem architecture	If you select 2oo3 or 2oo4 architecture for a sensor or final element subsystem, all groups within that subsystem must be identical (this is a SILability restriction).	If you are not sure, set all the group's "Safe failures repaired within MTTR" checkboxes to unchecked. (See Chapter 6 for more details.)
Selected architecture can't be used	You selected an architecture that SILability can't calculate.	Assign a different architecture in the affected element.
Selected architecture doesn't match number of groups defined	In a subsystem, the *MooN* architecture must match the number of groups plus sub-SIFs (*N*).	Select an architecture *MooN* where *N* is the same as the number of groups plus sub-SIFs. You may have accidentally added an extra empty group to the subsystem. If the subsystem contains empty or unneeded groups, delete them.

Message	Explanation	Possible solutions
Selected architecture doesn't match number of legs defined	In a group, the *MooN* architecture must match the number of legs (*N*).	Select an architecture *MooN* where *N* is the same as the number of legs. You may have accidentally added an extra empty leg to the group. If the group contains empty or unneeded legs, delete them.
Subsystem β can't be greater than group β	In a subsystem, SILability assumes that common cause failure between legs within a group is more likely than between groups, as legs will generally have relatively more common failure modes. Therefore, the β of the subsystem should not be greater than the β of any group in the subsystem.	Increase group β, or reduce subsystem β. If you see this message unexpectedly, you may have set mismatched units for β (absolute value, such as 0.1, or percentage, such as 10%).
Subsystem architecture must be selected	In each subsystem that is included in the SIF, you must specify the architecture to be used in the calculation.	Set the architecture to any value *MooN* where *N* is the number of groups plus sub-SIFs in the subsystem.
Type B legs must have DC ≥ 0.6 in Route 2H AC model	In IEC 61508 AC route 2H, there is a requirement that all type B elements have diagnostic coverage ≥ 0.6. This message means that SILability has calculated the DC of a leg containing one or more type B component(s) as < 0.6.	Check the component λ values in the leg are correct. Check that the component AC type (A or B) is set correctly for all components in the leg. Use a different AC model (this is set at SIF level).
Use λ high/low checkbox must be defined when leg's trip direction/action is defined	If you set the trip action/direction of a leg to any value other than "Undefined", you need to tell SILability whether you want to use λ high/low/freeze values rather than λ Total values for the components in the leg.	• Set the "Use λ high/low" checkbox to either checked or unchecked *or* • Set the leg's "Trip action" to "Undefined". This will make SILability use λTotal values for the components in the leg.

Message	Explanation	Possible solutions
'Use PVST' checkbox must be defined	In every final element leg, you must specify whether to use PVST.	If you are not sure, set the leg's "Use PVST" checkbox to unchecked.
Value is outside acceptable range	A numerical value is outside the allowable range.	If the value seems to be correct, check that you have selected the appropriate unit. For example, you may have selected hr^{-1} instead of FIT for a λ value. (You can enter values in any unit; SILability automatically converts the values to the units required for calculation.)
Value is undefined	A required numerical parameter has not been assigned any value. It will be shown as "Not set" in the data entry panel.	This may occur if: • You left a 'dead' element (e.g. component) in the SIF that you don't intend to use. Delete the element and change the SIF architecture to match (eg from 1oo3 to 1oo2). *or* • SILability is looking for a value that you didn't realise was required. For example, if you select "Use λ high/low" in a component, you need to supply values for λD/U High/Low. Unselect the "Use λ high/low" checkbox to remove this requirement.
Values must be identical between groups when using this subsystem architecture	Refer to "All component AC types must be identical when using this group/subsystem architecture".	
λ high/low/-freeze values needed	In a component, if you select "Use λ high/low" and set "Trip action" to any value other than "Undefined", you need to supply values for λD/U High, Low and Freeze.	Remove this requirement by: • unselecting the "Use λ high/low" checkbox; *or* • setting "Trip action" to "Undefined"

Message	Explanation	Possible solutions
λDD/DU/SD/SU Total values needed	In a component, if you deselect “Use λ high/low” or set “Trip action” to “Undefined”, you need to supply values for λDD, DU, SD and SU Total.	If you do not have values for λ_{DD} or λ_{SD} (because the component has no diagnostics), set these values to zero. If you want to use λ high/low values instead, select the “Use λ high/low” checkbox, and set “Trip action” to any value other than “Undefined”.

9. CHAPTER 9 – GENERATING REPORTS AND EXPORTS

9.1 INTRODUCTION

When you have completed a SIL verification task, it's likely you will need to produce a report. SILability supports this by providing several reporting and exporting functions:

- **Full report** of all input data used and all calculation results achieved for each SIF, with a table showing project information. This is a nicely formatted spreadsheet in Microsoft Excel format (.xlsx) containing tables showing input data and calculation results. You can easily paste the spreadsheet contents into a word processor as part of your SIL verification report. The spreadsheet column widths are preset to suit a typical A4 portrait layout document.
- **List of all SIFs** in the project, with a summary table showing how many SIFs achieved their risk reduction targets. This is also a pre-formatted Microsoft Excel document.
- **Architecture diagram**, showing the architecture of some or all SIFs in the project as .png images.
- **Export of SIF data**. This is a complete export of all the raw input data and calculated results for all SIFs in the project, as a Microsoft Excel (.xlsx) file.

9.2 HOW TO GENERATE A REPORT OR EXPORT

In the File menu, select the type of report or export you require. The data entry panel (top right of the screen) shows applicable options as follows:

- Filename: The full filename for the spreadsheet document to be created.
- "Select" button: This opens a standard file name entry dialogue. The file name you select in the dialogue will be inserted into the "Filename" field.
- "Overwrite existing file" checkbox: if a file already exists with the filename specified, it will only be overwritten if you check this checkbox.
- "All SIFs in one worksheet" / "Separate worksheet per SIF" or "Summary and detail in one worksheet" / "Summary and detail in separate worksheets" checkboxes: Select the report style you require: either everything in one worksheet, or a separate worksheet (in the same spreadsheet file) per table.
- "Replace '%' in the filename with": if you are requesting an architecture diagram of more than one SIF, the diagrams will be created as separate files. To allow SILability to create a unique filename for each one, you need to insert a % sign into the filename. SILability will replace the % with either the SIF's tag number or its serial number for each SIF.

- “Run data checks” button: This button runs the data consistency checks described in Chapter 10. You don’t have to run the checks before producing the report, but it’s highly recommended.
- “Exit” button: Cancels the reporting operation and returns to the normal data entry display.
- “Go” button: Produces the report.

9.3 WHAT THE FULL REPORT CONTAINS

9.3.1 Input data

All input data provided by the user is shown in the full report. This includes the project's “edit number”, which, as explained in Chapter 3, is incremented every time the data is changed. This allows you to confirm that the data used to generate the report is exactly the same as the data in the latest version of the project file.

Any undefined (“Not set”) numerical values are shown as <Undefined>. Any undefined checkbox values are shown as Undefined.

Some input data fields are omitted from the report if they are not relevant. For example, in any system with 1oo1 architecture, the β (common cause) factor is not applicable and so it is not shown.

All comments entered by the user are shown at the bottom of each section of the report. Comments are numbered, and the corresponding numbers are shown in square brackets next to the corresponding data item, like this: [1 2] indicates there are two comments, no. 1 and no. 2. Identical comments within each section are assigned the same number to avoid repetition. (Comments attached to undefined boolean parameters are omitted from the report.)

The total number of SIFs in the project is displayed within the project overview section, excluding any SIFs which are used as sub-SIFs.

9.3.2 Outputs (Calculation results)

All calculated results are shown in the relevant sections. If any calculations could not be performed (as explained in Chapter 7), the results are shown as “not evaluated”.

9.4 WHAT THE SIF LIST REPORT CONTAINS

- A summary table showing how many SIFs are in the project (including sub-SIFs), and how many SIFs achieved their risk reduction target. The results are broken down per SIL.
- A SIF list table showing the overall results for each SIF, and whether it achieved its risk reduction target. SIFs that could not be evaluated are shown as <ND> (not determined).

10. CHAPTER 10 - SPECIAL TOOLS AND TIPS

10.1 DATA CHECKS

The task of SIL verification requires a huge number of input data items, derived from many different sources. No matter how carefully the data is entered, it is easy to make a mistake while entering data, or to forget that a change to the data in one SIF may need to be mirrored in another SIF.

To help the functional safety engineer work towards an error-free SIL verification, SILability provides a powerful data check tool. This runs an array of consistency and reasonableness checks on the entire data set within a project, and reports any anomalies found.

You can run the data checks by selecting the Tools menu and clicking Data checks, or by pressing the “Data check” button in the spreadsheet output setup screen (see Chapter 8).

Anomalies found will be reported in the result panel at the bottom right of the screen. All the anomalies in the entire project will be shown at once, not just the anomalies relating to the currently selected SIF and element (in the tree view). If several anomalies are found, you may need to use the scroll bar to view the complete list. Each item reported has a “Show me” button that navigates to the affected data item, just like the problem report described in Chapter 7.

Checks on specific data items are skipped if the data items are not needed in the calculation.

Please note, even if anomalies are found, SILability will still attempt to perform the calculations. You are strongly advised to run the checks immediately before producing a report. It is wise to run the checks at other times as well, as you progress through the project, to help you catch any data entry problems as early as possible.

A complete list of data checks performed by SILability is shown below. Details of the meaning of each data item are given in Chapter 6.

10.1.1 SIF level

- AC model: Consistent for all SIFs across the project
- PFD/RRF target: The user's choice of whether to provide a PFDavg target or an RRF target is consistent for all SIFs across the project
- AC model: Not using AC model IEC 61511:2003 in combination with high demand operating mode. High demand mode is defined only in IEC 61508 and in IEC 61511:2016, not in IEC 61511:2003, so this would mean that the user is applying inconsistent standards.

10.1.2 Component level

- Proof test coverage: Not less than diagnostic coverage for the component
- Proof test coverage: Consistent across similar components across the project
- PVST coverage: Not more than proof test coverage for the component
- PVST coverage: Consistent across similar components across the project
- β factor: In the range 2~10%
- λ values: In the range 0~10 000 FIT
- λ values: Consistent across similar components across the project
- Proven in use (checkbox): Consistent across similar components across the project
- Use λ high/low (checkbox): Consistent for components with the same λ values
- Diagnostics are frequent enough (checkbox): Consistent for components with the same λ values
- Parameter adjustment is limited (checkbox): Consistent for components with the same λ values

10.1.3 Leg level

- Use PVST (checkbox): Consistent across similar legs across the project

10.1.4 Group level

- Proof test interval: Not less than PVST interval for the group
- Proof test interval: Not more than mission time for the group
- Proof test success rate: In the range 80~100%
- Proof test success rate: Consistent for similar groups across the project
- Mission time: In the range 5~20 years
- Mission time: Consistent for similar groups across the project
- MTTR: In the range 1 hour ~ 6 months
- MTTR: Consistent for similar groups across the project
- PVST interval: At least 0.1 month
- PVST interval: Not more than proof test interval for the group

- PVST interval: Consistent for similar groups across the project
- PVST success rate: In the range 80~100%
- PVST success rate: Consistent for similar groups across the project
- β factor: In the range 2~10%
- β factor: Consistent for similar groups across the project
- MTTR includes diagnostics time (checkbox): Consistent for similar groups across the project
- Safe failures repaired within MTTR (checkbox): Consistent for similar groups across the project

10.1.5 Sensor and final element subsystem level

- Reduce HFT requirement (checkbox): Consistent for all Type A subsystems across the project, when using applicable AC models (IEC 61508:2010 Route 2H and IEC 61511:2016)

10.1.6 Logic solver subsystem level

- Detect safe failures in sensors (checkbox): Consistent for similar logic solvers across the project
- Programmable (checkbox): Consistent for similar logic solvers across the project
- Trip action (choice box): Consistent for similar logic solvers across the project
- MTTR includes diagnostics time (choice box): Consistent for similar logic solvers across the project
- Safe failures repaired within MTTR (checkbox): Consistent for similar logic solvers across the project
- Proof test interval: Not more than mission time for the subsystem
- Proof test success rate: In the range 80~100%
- Proof test success rate: Consistent for similar logic solvers across the project
- Mission time: In the range 5~20 years
- Mission time: Consistent for similar logic solvers across the project
- MTTR: In the range 1 hour ~ 6 months
- MTTR: Consistent for similar logic solvers across the project
- β factor: In the range 2~10%
- β factor: Consistent for similar logic solvers across the project
- Reduce HFT requirement (checkbox): Consistent for non-programmable logic solvers across the project, when using applicable AC models (IEC 61508:2010 Route 2H and IEC 61511:2016)

10.2 SIL VERIFICATION MODELLING TIPS

10.2.1 Modelling SIFs with no sensor components

Sometimes, the action of one SIF may directly place demand on another SIF, to prevent a secondary consequence arising as a result of the first SIF tripping. For example, if a centrifugal pump is tripped by a SIF, it may be necessary to close the pump's outlet valve to prevent backflow through the pump.

In this situation, the second SIF is known as a *secondary SIF* and should be modelled with no sensor components, as it is triggered directly by the first SIF via a software signal in the logic solver. This can be done by excluding the sensor subsystem from the SIF (uncheck the 'Include this subsystem in SIF' checkbox in the sensor's data entry panel).

Alternatively, you can model it as a sensor subsystem with no sensor components. Since SILability requires you to have at least one component in each included subsystem, you need to create a "dummy" component having zero failure rate values. You should also set the group's proof test duration to zero; this is needed because SILability allows for the probability that the group is unavailable due to testing, which is irrelevant if there are no components to test.

APPENDICES

APPENDIX A: WHERE TO GET FAILURE RATE DATA

The result of the SIL verification calculation depends heavily on valid failure rate data for the hardware components in the SIFs. Sometimes, published data is available for a specific device that matches or closely resembles the component you are using. In other cases, you may need to use generic data for a similar device, or an average obtained from several similar devices.

The best source of failure rate data will generally be the safety manual or SIL certificate for the exact component you are using. However, it is wise to be skeptical of very low λ_D claimed failure rates in low demand mode applications, especially if the λ_D is much lower than similar devices from other manufacturers. Some failure rate values are obtained by performing laboratory cycle tests, and this may give an optimistically low λ_D value compared with field (observed real-world) failure rates for devices with moving parts that stand idle for long periods of time. For similar reasons, be wary of applying B10 failure rates (which apply to continuous mode of operation) to low and high demand mode SIFs.

Sources of failure rate data

- *Guidelines for process equipment reliability data*, A CCPS publication from the American Institute of Chemical Engineers, 1989. This contains tables of generic failure rates measured in the field for a wide range of devices. In many cases, only a single failure rate is given, not broken down into types of failure, so it will generally only provide a worst-case λ_{DU}, not other λ values.
- *Offshore and onshore reliability data (OREDA)*, 6th edition. Mainly focuses on offshore equipment, but some onshore equipment is also included. Failure rates are collected from observed reliability in the field.
- *Safety automation equipment list*, online search at www.exida.com/SAEL. Contains FMEDA reports and SIL certificates, showing failure rate data, for a carefully selected range of hardware suitable for use in SIL-rated applications.
- *SILSafe data*, a collection of typical dangerous failure rates for safety-related hardware, www.silsafedata.com
- *Safety equipment reliability handbook,* 4th edition, exida. A collection of failure rate data assembled by exida from industry sources and from its own FMEDA studies. Breaks down the data to a fine level of detail, showing separate λ values for many individual devices.
- Individual equipment manufacturers. Some manufacturers make safety manuals available on their websites for download; these may include failure rate data and other necessary information such as proof test coverage. Other manufacturers may provide such data on request.

- Field failure measurements. If your organisation has a significant number of similar devices in service and keeps good failure event records, these records can be used to estimate real-world (or at least worst-case) failure rate values. The xSeriCon website, www.xsericon.world, has additional resources on this topic.

See also the resources listed in Appendix G for further databases and lists of failure rates.

APPENDIX B: ASSUMPTIONS MADE IN CALCULATION

The following assumptions are made in the algorithm used to perform SIL verification calculations in SILability.

1. All component failure rates are independent of time, throughout the mission time of the group to which the component belongs.
2. All components are in service only during their published lifetime, and will be fully refurbished or replaced at the end of their lifetime. “Infant mortality” (early failure of components due to manufacturing or material defects) are not considered, as these should be found by manufacturers during burn-in testing before components are shipped to end users.
3. All failures detectable by diagnostics are also discoverable by proof testing.
4. All failures detectable by partial valve stroke testing are also discoverable by full proof testing.
5. Proof testing execution is imperfect and may not be correctly executed in every case.
6. The proof test coverage, PVST coverage and β (common cause) factor are constant for all types of λ value (DD, DU, SD etc.) in a single component.
7. The β (common cause) factor is constant for all legs in a single group.
8. The β (common cause) factor is constant for all groups in a single subsystem.
9. The diagnostic interval is so short that the time waiting for a detectable fault to be detected is negligible in low demand and high demand modes.
10. No unprotected bypasses are used during proof tests and PVST. That is, the SIF will be able to detect the hazard (in the absence of faults) on completion of the proof test or PVST.
11. Proof test interval and PVST interval the same for all the legs within a group.
12. For a *M*oo*N* architecture in a group where $1<M<N$, all the legs in the group are required to be identical. The same applies to groups within a sensor subsystem.
13. The downtime of a SIF due to restoration following discovery of simultaneous non-common cause failures is negligible in low demand mode.
14. MTTR is constant for all legs within a group.
15. In a logic solver subsystem with architecture other than 1oo1, the channels of the logic solver are identical.
16. Only one standard, IEC 61508:2010, IEC 61511:2003 or IEC 61511:2016, is applied to each SIF. If IEC 61508:2010 is applied, either Route 1H or Route 2H is selected for the whole SIF.

17. If a subsystem contains redundant legs or groups, the legs/groups are not tested simultaneously, so that the equipment under control remains protected during proof testing.
18. The probability of failure of a redundant system does not change during proof testing and restoration, even though the effective architecture is different at that time.
19. For a SIF protecting against a hazard that is not always present, the fraction of time that the hazard is present (sometime called *usage ratio*) is not taken into account. It is assumed that credit has already been taken for this at the SIL determination stage.
20. Detected safe failures are restored during the MTTR.
21. There is no redundancy between different kinds of components in a single leg. That is, the architecture of a leg is assumed to be *N*oo*N*.
22. The components within a 1oo2 component architecture are identical.
23. Any group or logic solver reaching the end of its mission time is fully restored immediately.
24. IEC 61508:2010-2 Clause 7.4.4.1.4 (taking credit for diagnostics) is applied only to SFF calculation and not PFD/PFH calculations.

APPENDIX C: KNOWN LIMITATIONS

The current version of SILability is known to have the following issues, which will be addressed in future versions:

1. Although it is possible to open a SILability project file created using third party software, SILability does not robustly check that the file is valid. This may lead to a software crash.
2. The main screen can be difficult to use on low resolution screens (800 x 600).
3. If you have made a large number of edits since the project file was last opened, the 'save on the fly' file may contain many edit records, which can slow the SIF calculation. Close and reopen the project file occasionally to resolve this.
4. If you use a component with a very high Diagnostic Coverage (high λ_{DD}: λ_{DU} ratio) in low demand mode, SILability may refuse to calculate the PFDavg of the component because the effective proof test coverage is less than the Diagnostic Coverage. The workaround is to set the proof test coverage and proof test success rate (of the group containing the component) to 100%. This will not affect the component in question (because it has essentially no discoverable faults). However, the high proof test success rate will affect other components in the group; therefore, you may need to decrease the proof test coverage of these components, so that the effective coverage (calculated as proof test coverage x proof test success rate) is unchanged. Alternatively, move the other components into a different group.

APPENDIX D: FORMAT OF XML PROJECT FILE

Project and database files created and opened by SILability are in XML format. Please contact xSeriCon if you require details of the XML syntax.

APPENDIX E: TERMS & ABBREVIATIONS

Term or Abbreviation	Definition
β (Beta)	Common cause factor: the fraction of faults (safe and dangerous) that would affect all redundant elements simultaneously
λ (Lambda)	Failure rate
λ_{DD}	Dangerous failure rate detectable by diagnostics
λ_{DU}	Dangerous failure rate undetectable by diagnostics
λ_{SD}	Safe failure rate detectable by diagnostics
λ_{SU}	Safe failure rate undetectable by diagnostics
AC	Architectural constraints
C&ED	Cause and effect diagram
CPU	Central processing unit
DC	Diagnostic coverage
ESD	Emergency shutdown
EUC	Equipment under control
FSA	Functional Safety Assessment – an activity in which a SIS is assessed to determine if it achieves its safety objectives
HAZOP	Hazard and operability (study)
HFT	Hardware fault tolerance
I/O	Input/output
IEC	International Electrotechnical Commission
IPF	Instrumented protection (or protective) function
IS	Intrinsically safe
macOS	The operating system provided with Apple Mac computers (starting from macOS 10.12)
*M*oo*N*	An architecture with *N* voters, such that at least *M* votes are required to generate a trip
MTTFS	Mean time to fail safe
MTTR	Mean time to restore
*N*oo*N*	An architecture with *N* voters, such that all *N* votes are required to generate a trip
P&ID	Piping and instrumentation diagram
PF	Probability of failure

PFD	Probability of failure on demand
PFDavg	Probability on demand, averaged over the lifetime of the project
PFH	Probability of failure per hour
PLC	Programmable logic controller
PIU	Proven in use
PT	Proof test
PTC	Proof test coverage
PTI	Proof test interval
PVST	Partial valve stroke test
Route 1H	Method within IEC 61508:2010 to determine the hardware fault tolerance requirement
Route 2H	Method within IEC 61508:2010 to determine the hardware fault tolerance requirement
RTD	Resistance temperature detector
RRF	Risk reduction factor
SFF	Safe failure fraction (the ratio of the rates of safe failures to safe + dangerous failures in a component or assembly of components)
SIL	Safety integrity level
SIF	Safety instrumented function
SIS	Safety instrumented system
SRS	Safety requirements specification
XML	Extensible Markup Language

APPENDIX F: TERMS & CONDITIONS

Acceptance of Terms

xSeriCon Limited (“Company”) offers the use and services of SILability software (“Software”) to customer/client (“User”) subject to the following terms and conditions (“Terms of Use”), which may be modified from time to time without prior notice. By continuing to use this software following such changes, you agree to be bound by such modifications.

Authorized User

In regards to paid Company products and services, User agrees not to provide or make known his/her product key to any other person to enable that person’s access and unauthorized use of the Software. The User’s computer used to register the Software is the only computer licensed to use the product key supplied by Company. User agrees to provide true, accurate, current and complete information as requested by Company and maintain and promptly update the registration data to keep it true, accurate, current and complete.

Rights to Make Changes

The Company reserves the right to change or discontinue, temporarily or permanently, the use of Software (or any part thereof) at any time without prior notice. The Company shall not be liable to the User or to any third party for any modification, suspension or discontinuance of the Software.

Links to other websites

The Company shall not be responsible for the contents available on or the set-up of any other websites linked to this Software. Access to and use of such other websites is at the User’s own risk and subject to any terms and conditions applicable to such access/use. By providing hyperlinks to other websites, if any, the Company shall not be deemed to endorse, recommend, approve, guarantee or introduce any third parties or the service/products they provide on their web site, or have any form of cooperation with such third parties and websites. The Company is not a party to any contractual arrangements entered into between the User and the provider of the external website.

Links to SILability Website

User may set up a hyperlink to this Software's website by first obtaining the written approval from the Company (which may be withdrawn at any time at the discretion of the Company). User will terminate the hyperlink within two days of receipt of a notice from the Company.

User Conduct

As a condition of User's use of this Software, User must not:
a. trespass, break into, access, use or attempt to trespass, break into, access or use any other parts of Company servers, and/or any data areas for which the User has not been authorised by Company;
b. restrict or inhibit any other licensed User from using and enjoying this Software.

Intellectual property rights

All intellectual property rights subsisting in respect of this Software belong to the Company. Except with the express permission of the Company, Users are not allowed to upload, post, publish, reproduce, transmit or distribute in any way any component of this Software itself or create derivative works with respect thereto. User agrees that the Company are free to use, disclose, adopt and modify all and any ideas, concepts, knowhow, proposals, suggestions, comments and other communications and information ("Feedback") provided by User to the Company in connection with its Software and/or products and services without any payment to the User. User hereby waives and agrees to waive all and any rights and claims for any consideration, fees, royalties, charges and/or other payments in relation to Company use, disclosure, adoption and/or modification of any or all of User Feedback.

Indemnity

User agrees to indemnify, defend and hold harmless the Company from and against all liabilities, claims, actions, costs, expenses, loss and damages arising or in connection with the User's breach of the Terms of Use and/or any other activity by the User in connection with the use of this Software.

In no event shall the Company be liable to any party for direct, indirect, special, incidental, or consequential damages, including lost profits, arising out of the use of this software and its documentation, even if the Company has been advised of the possibility of such damage.

The Software and accompanying documentation, if any, provided hereunder is provided “as is”. The Company has no obligation to provide maintenance, support, updates, enhancements, or modifications.

Termination

The Company may in its sole discretion, terminate or suspend the User’s access to all or part of the Software for any reason, including, without limitation, breach of the Terms of Use. The Company will not be liable to the User or any third party for any claims related to the termination of the use of Software.

Privacy Policy

For information about Company privacy policies and practices, please refer to Company ‘Privacy Policy Statement’ below.

Governing Law and Jurisdiction

The Terms of Use shall be governed by the law of the Hong Kong Special Administrative Region. User agrees to submit to the non-exclusive jurisdiction of the Hong Kong courts.

Language Version

If there is any conflict or difference between the different language versions of the ‘Terms of Use’ and its English version, the English version prevails. If any part of the other language version is unclear, reference should be made to the English version. The language version may not be translated into English for the purpose of comparing with or interpreting the English version.

PRIVACY POLICY STATEMENT

xSeriCon Limited ("the Company") is committed to protecting User privacy.

Types of Data Collected

From time to time, it is or will be necessary for the User to supply to the Company data including but not limited to when the User has enquiries or purchases Company products & services. Visiting Company websites may place "cookies" in the User's browser to collect personal identification data such as name, date of birth, email address, address, telephone number, account number and other relevant information to realize User preferences. Failure to supply such data may result in the Company's being unable to provide its products & services to the User.

Purposes for which data are used

The purposes for which the Company may use the data collected or to be collected by the Company are divided into obligatory purposes and voluntary purposes. If the data is to be used for an obligatory purpose, User MUST provide the data to the Company to provide the relevant products or services for which the User has requested. If the data is only to be used for a voluntary purpose, the Company will obtain User consent and will have the option to tell the Company not to use the data for that purpose and we will not do so.

Purposes for which it is obligatory for you to provide the data are:

1. handling and following up enquiries and matters related hereto;
2. meeting the requirements to make disclosure under requirements of any law binding on the Company;
3. to process User requests or enquiries;
4. to process and complete transaction(s) requested by User;
5. designing new or enhancing existing products & services provided by Company for User's use;
6. to send User administrative communications, such as information about any account User may have with the Company or about future changes to this Privacy Policy Statement;
7. to administer and enforce the rules of promotions and/or the terms of Company commercial dealings;
8. for Company internal business and administrative purposes;
9. to assist in law enforcement purposes and to meet requirements imposed by law or for

claims- related purposes;
10. for safety or security purposes;
11. to ensure ongoing creditworthiness of the User;
12. to determine amounts owed to or by the User;
13. to enforce User obligations, including without limitation the collection of amounts outstanding from User and those providing security for User;
14. purposes relating thereto.

Purposes for which it is voluntary for User to provide the data are:
1. sending to User direct marketing, promotional information and/or materials and/or offers and news of the Company's products and services,
2. sending User our future marketing purposes and in conjunction with products; and
3. conducting customer and service surveys.

Notice for Direct Marketing

Company intends to use User personal data including your name, telephone numbers, fax number, email and other contact information collected for marketing communications in relation to the classes of products and services that are on offer.
Company will not use your personal information for direct marketing without User consent. If User does not allow and agree the Company to use the data for the voluntary purposes including direct marketing as listed above, please inform Company by using the opt-out method below.

You may opt-out from receiving any of the Company's direct marketing information and services at any time, free of charge. To opt-out please email: info@xsericon.world together with your name and invoice number.

User will be removed from Company mailing list within fourteen (14) Hong Kong business days upon Company's receipt of User's request.

Disclosure of Personal Data

Data held by the Company relating to the User will be kept confidential. In cases where the Company does collect the data from the User, we will:
1. notify User (by way of this Privacy Policy Statement or by a separate notification) that the Company is doing so and the use that the Company will make of such data we collect;
2. where relevant, the Company will give User the opportunity to "opt out" (that is to restrict the uses the Company will make of such data);

The Company may, where such disclosure is necessary to satisfy the purpose, or a directly related purpose, for which the data was collected provide such data to the following parties:

1. any agent, contractor or third party service provider who provides services to the Company;
2. any credit reference agencies, in the event of default, to debt collection agencies; and
3. any other person or company who is under a duty of confidentiality to the Company and has undertaken to keep such information confidential.

Access and Correction of Personal Data

User has the right to:
1. check whether the Company holds any personal data relating to the User and the right of access to such data;
2. require the Company to correct any data relating to the User which is inaccurate; and
3. ascertain the Company's policies and practices in relation to personal data.

Request for access and correction of data and the kinds of data held should be emailed to Company addressed to info@xsericon.world together with your name and invoice number in the subject line.

You may also request the Company to delete the data from any active mailing or distribution list. To exercise any of the User’s rights, contact us by emailing info@xsericon.world.

The Company may take reasonable steps to verify User identity before granting access or permitting corrections to User information. The Company has the right to charge a reasonable fee for the processing of any data access request.

APPENDIX G: FURTHER READING

If you are not familiar with the relevant IEC standards (as listed below) and the theory and practice of SIL verification in the context of functional safety engineering, you are strongly advised to familiarise yourself with the following materials before using SILability. xSeriCon provides training in these areas in online and classroom formats; please enquire for details.

1. International Electrotechnical Commission (IEC), 2016, IEC 61511 Functional safety—safety instrumented systems for the process industry sector, Parts 1 and 2.
2. International Electrotechnical Commission (IEC), 2010, IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems, Parts 1, 2, 4 and 7.

3. I. Cameron and R. Raman, *Process systems risk management,* Elsevier, 2005. Section 8.7.2 provides a substantial list of sources of failure rate data.
4. W.M. Goble and H. Cheddie, *Safety instrumented systems verification: practical probabilistic calculations*, ISA, 2005. The standard textbook on the theory of SIL verification.
5. W.M. Goble, *Control systems safety evaluation and reliability*, 3rd edition, ISA, 2010. Important background on the failure behaviour of safety-related equipment.
6. K.J. Kirkcaldy and D. Chauhan, *Functional safety in the process industry*, self-published (available from Amazon), 2012. Chapters 13 and 14 contain a gentle introduction to SIL verification and the calculation of failure rates.
7. D.J. Smith and K.G.L. Simpson, *Safety critical systems handbook*, 3rd edition, Butterworth-Heinemann, 2011. Chapter 6 is dedicated to the sourcing of failure rate values.

APPENDIX H: TROUBLESHOOTING AND FAQ'S

If you experience problems running SILability, please check the questions and answers below. Should you still have unresolved issues, please contact the SILability team at xSeriCon, who will assist you as quickly as possible.

Licensing

	Question	Answer
1	After updating macOS on my Mac computer, my SILability license doesn't work. I see the message "Your computer seems to have changed".	An OS update may change the ID code used by the SILability licensing server to identify your computer. If you are planning a MacOS update, contact us in advance and at a time of your choosing we can disassociate your license key with the old OS so that you can access SILability after your OS update.
2	I have a valid SILability license key, but SILability is not recognising it.	A license key is valid for one computer only. If a key has already been used on one computer, it can't be used on another. If your computer has become unusable, contact xSeriCon for assistance. License keys have an expiry date. Please contact xSeriCon to find out if your license has expired. The license key must be entered exactly as supplied by xSeriCon. It is case-sensitive. Please try pasting the key from the email you received from xSeriCon. If you are retyping it, check you are not confusing letters I/O with digits 1/0.

	Question	Answer
3	On launch, SILability shows this message: “We’re sorry, there’s a problem with the licensing server. Try launching SILability again”.	This indicates that the licensing server didn’t respond. Your computer, organisation or ISP may have a firewall that’s blocking the licensing server. It is less likely but possible that the server may be down: please contact xSeriCon.
4	On launch, SILability shows this message: “SILability hit a compatibility problem. Please contact xSeriCon for assistance”.	In rare cases, an OS version incompatibility means SILability can’t use a software library needed to read your software license. Please contact xSeriCon.
5	My SIL verification project is finished and I don’t need the SILability license any more. Can I get a refund?	We’re sorry, but licenses are not refundable once activated.
6	When I first run SILability, my firewall warns me that SILability is trying to access http://8.8.8.8. What is this, and is it safe?	This is the address of a public DNS (domain name server) run by Google. It is called to confirm that your computer has internet access, so that it can reach xSeriCon’s licensing server. No data will be downloaded from 8.8.8.8 and you can safely allow access.

Entering data

	Question	Answer
7	I would like to perform a quick and simple calculation just by entering λ_{DU} values. Why do I have to enter several other parameters as well?	SILability intentionally doesn't provide default values for any parameter, to avoid inadvertent values getting used in the calculation. You can follow the guide in the result panel (bottom right of the screen) to show you which parameters must be entered. If you frequently use the same basic model with different values of λ_{DU}, you can save it as a project. Consider using SILability's sandbox function.
8	Can I enter a PFDavg value instead of a λ_{DU} value for a component?	No, SILability doesn't accept direct input of PFDavg values, because they depend on several assumptions that must be verified. It is usually possible to back-calculate λ_{DU} from PFDavg values provided in literature; contact xSeriCon for guidance.
9	Why is proof test success rate entered at group level, yet proof test coverage entered at component level?	We expect that proof test success rate (a performance measure of the quality of maintenance) will not vary from one component to another. On the other hand, proof test coverage will vary for each component in the SIF.
10	How can I change the order of groups, legs and components? And how can I move a leg from one group to another, or move a component from one leg to another?	Use Element menu – Copy element to another SIF to copy the element you want to move at its new location (it works even in the same SIF). Then use the Element menu – Delete element command to delete the original instance of the element.
11	Some of the data input fields are not visible in the data entry panel (top right area of the screen).	You may need to drag the horizontal divider between the data entry panel and result panel downwards to reveal the missing fields.
12	I can't see the components of my SIF in the tree view (left side of the screen).	By default, legs are shown 'collapsed' in the tree view to save space. Expand the legs by clicking the small triangle on the left of the leg.

File handling and database

	Question	Answer
13	My computer crashed while I was using SILability. Have I lost all my work on the current project?	If you were using 'save on the fly', nothing should be lost. Simply reopen the project file in SILability and all your work should be recovered automatically. We highly recommend using 'save on the fly' at all times. To use it, choose Save As from the File menu, as soon as you open a project file. If you weren't using save on the fly, then unfortunately your work since the last manual save is lost. The same applies to any changes made to an open user database.
14	Why doesn't the number of SIFs in the project match the number of SIFs in the project overview section of the spreadsheet detailed report?	Sub-SIFs are not included within the "Number of SIFs" field of the project overview section of the spreadsheet detailed report.
15	I am opening a database that I know contains discrepancies relative to components in the currently open project. Yet I am not seeing a discrepancy warning. Why is this?	If any components in your project have their "Override database" checkboxes checked, the discrepancy check will not be performed for these components.
16	I am resolving discrepancies when opening a database. When I select "overwrite database value", a new discrepancy appears. Why is this?	"Overwrite database value" changes the component parameters in the database. This might cause a new discrepancy with another component in the project. SILability is showing you the new discrepancy resulting from the change.
17	The SILability user guide states that all user-entered comments show up in the Excel report. Are there any exceptions to this?	Comments attached to checkboxes that are not set (neither checked nor unchecked) are not saved in the project file. As a result, they will not be shown in the report when the project is reopened.

	Question	Answer
18	I need a full export of all modelling parameters in a spreadsheet. Can SILability provide this?	Yes. Request a %%%SIF list export (see section 9.2). SILability will also produce a separate Microsoft Excel file containing a full dump of all parameters for all SIFs in your project.

Modelling issues

	Question	Answer
19	My sensor subsystem contains just one set of transmitters in a 2oo3 architecture. Should I model it as three legs in one group, or three groups containing one leg each?	Both methods are acceptable, and they will give essentially the same result. xSeriCon recommends modelling this as three legs in one group, as it will be easier to enter the data.
20	Where can I find λ values (failure rates) to use in my SIL verification project?	Please see Appendices A and G of the user guide for guidance.
21	The component or leg shows a lower than expected safe failure fraction. What's the cause?	In low demand mode: Go to the corresponding group, and check whether the "MTTR includes diagnostics time" checkbox is checked. In high demand or continuous mode: Go to the corresponding components, and check whether the "Diagnostics are frequent enough" checkboxes are checked. In each case, if the checkboxes are not checked, λ_{DD} failures will be treated as dangerous failures, leading to a lower SFF.

	Question	Answer
22	Why does my probability of failure increase when I remove a component / decrease its dangerous failure rate?	In a *N*oo*N* architecture it is possible that a reduction in the probability of failure of a leg, via change of component failure rates or removal of a component, can cause an increase in the overall probability of failure of a group. This is due to a reduction in common cause probability of failure and the dominance of the non-common cause failure rate. This effect may also occur at subsystem level after a change to one of the groups within a *N*oo*N* configuration.
23	Why does my PFDavg/PFH increase for a subsystem after disabling another subsystem?	This can occur if the subsystem you have disabled contains the longest mission time in the SIF. In SILability low demand mode, PFD values are calculated for operating times up to the longest mission time in the SIF. When each group reaches its mission time, it is assumed to be returned to as-new condition, resulting in lower PFD values for times after its mission time until the time limit is reached. When these lower PFD values are no longer included in the calculation, the PFDavg will increase.
24	Why does the MTTFS of my sensor subsystem decrease after disabling the logic solver subsystem?	The 'detect safe failures' and 'programmable' switches in the logic solver subsystem may have been checked previously. Disabling the logic solver will automatically treat these switches as unchecked, because the logic solver is not available to detect safe failures. This will cause the MTTFS of the sensor subsystem to decrease.
25	When estimating MTTR, should I include the time during which the process is offline, if I have to shut down the process to repair the SIF?	MTTR represents the time during which the hazard is not addressed by the SIF due to the SIF being offline. It does not have to include time during which the hazard is not present. If the process is taken offline for some time *T* during the repair, such that the hazard addressed by the SIF is not present, the MTTR can be reduced by *T*.
26	My group has redundant legs (e.g. 2oo3). If one leg fails, the group can still address the hazard. Should I therefore set the group's MTTR to zero?	No, you should still set the MTTR as if no redundancy is present. The MTTR is required for calculation of MTTFS in certain cases. SILability will automatically take redundancy into account.

	Question	Answer
27	My SIF had a sub-SIF in it, but now it has disappeared. As a result, the subsystem architecture no longer matches the number of groups. What happened?	You may have deleted the SIF that was used as the sub-SIF. When you do this, SILability automatically deletes it from any SIFs hosting it as a sub-SIF. You can undo the SIF deletion; the sub-SIFs will then reappear automatically. If you intentionally deleted the sub-SIF, you will have to manually adjust the architecture of all subsystems that were hosting it. SILability does not adjust the architecture automatically, to avoid making assumptions about the architecture you intended to use. To help you find SIFs that contain host sub-SIFs, use the SIF list function (Tools menu – Show SIF list) before deleting the sub-SIF.
28	When I select “Reduce HFT requirement” for a SIF using IEC 61511:2016 for a SIL 4 SIF, why is the minimum HFT required set to 1? This is not stated in the standard.	Since clause 11.4.6 of IEC 61511:2016 does not specify values to reduce the HFT requirements to, xSeriCon used clause 7.4.4.3.2 from IEC 61508:2010 Route 2H (which was where clause 11.4.6 was derived from) to select a minimum HFT of 1 for SIL 4.
29	How does SILability differentiate between FPL, LVL, and FVL for a component?	A FPL component will be AC type B and will have the “Parameter adjustment is protected” switch checked. An LVL or FVL component is AC type B and will have the “Parameter adjustment is protected” switch unchecked.
30	If I use a component with a very high Diagnostic Coverage (high λ_{DD}: λ_{DU} ratio) in low demand mode, SILability refuses to calculate the PFDavg of the component.	This happens whenever the effective proof test coverage is less than the Diagnostic Coverage (a physically unreasonable situation, as it means diagnostics find more faults than proof testing). The workaround is to set the proof test coverage and proof test success rate (of the group or logic solver containing the component) to 100%. This will not affect the component in question (because it has essentially no discoverable faults that require proof testing). However, the high proof test success rate will affect other components in the group or logic solver; therefore, you may need to decrease the proof test coverage of these components, so that the effective coverage (calculated as proof test coverage x proof test success rate) is unchanged.

	Question	Answer
31	My sensor subsystem contains only one component. The group PFDavg shows up higher than the component PFDavg. Why?	When calculating the total PFDavg for the complete subsystem, SILability calculates the sum of the unavailability contributions from the following: 1. Non-common cause dangerous failure of each component - as shown in the PFDavg for each component 2. Common cause dangerous failure of each component 3. Unavailability of the groups due to proof testing (i.e. the % of time the SIF is bypassed for testing) 4. Unavailability of the groups due to repair when a fault is detected (by proof testing or diagnostics) The PFDavg value shown per component takes only contribution 1 into account. But the PFDavg at group level includes all 1+2+3+4. In this case, the main difference is caused by 3, the proof testing. To confirm this, try changing the proof test interval (PTI) and proof test duration (PTD). As you decrease the PTI, or increase the PTD, the SIF spends more time under testing, so the group level PFDavg increases. If you go to the extreme case and set PTI = mission time (i.e. no proof testing in the life of the SIF), you will see the component and group PFDavg become equal.

Performance issues

32	Why is the recalculation of the SIFs gradually getting slower?	Refer to Appendix C for guidance on this issue.
33	Recalculation gets slower when I decrease PTI or PVSTI values below 3 months.	In order to optimize performance, SILability dynamically selects a sampling interval depending on the shortest PTI and PVSTI in the SIF. If you need to use short PTI or PVSTI values, one solution is to set all the other parameters of the SIF first, and then set the PTI/PVSTI at the end.