Instruments

Antidoc v3.0.0, Ron Dexter

Table of Contents

Chapter 1. Project description

No description found (add content in project description)

No description found (add content in project description)

Chapter 2. Libraries

This section describes the libraries contained in the project.

2.1. Instruments.lvlib

Responsibility: This is the top abstraction layer for any instrument. Descendent classes will need to adhere to the override settings, and add more information to the documentation where the overrides are implemented.

Version: 1.0.0.0

Table 1. Nested libraries

Name	Type
Measure Msg.lvclass	LVClass
Power Off Msg.lvclass	LVClass
Request Data Msg.lvclass	LVClass
Send Command Msg.lvclass	LVClass
Take Snapshot Msg.lvclass	LVClass
Instruments.lvclass	LVClass

2.1.1. Functions

This library has no functions set to non private scope.

2.1.2. Library Constant VIs

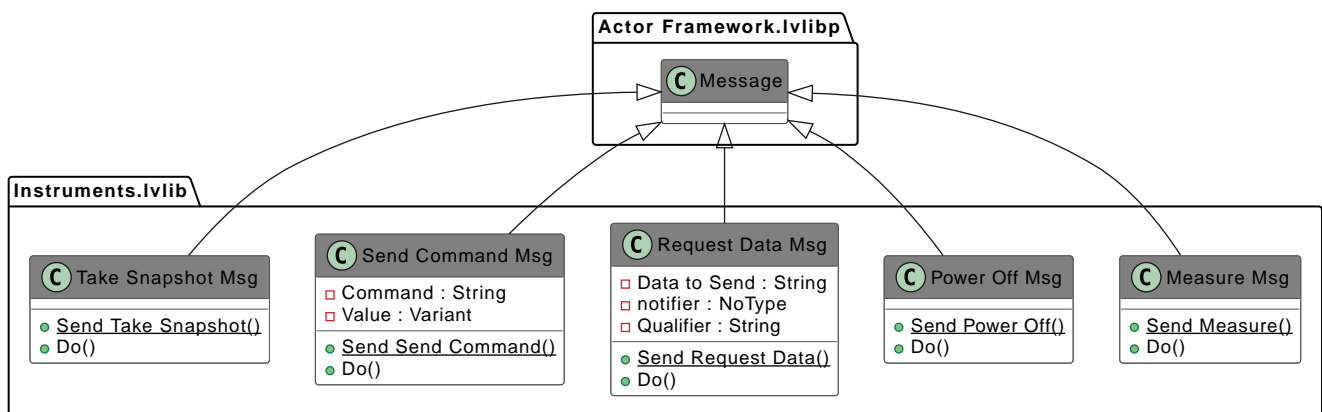| NOTE	No Constant VIs Found

Chapter 3. Classes

This section describes the classes contained in the project.

3.1. Classes overview

This project contains 5 classes and 0 interface.

Table 2. Classes list

| Classes	Interfaces
Measure Msg.lvclass	
Power Off Msg.lvclass	
Request Data Msg.lvclass	
Send Command Msg.lvclass	
Take Snapshot Msg.lvclass	

3.2. Measure Msg.lvclass

Responsibility: This is the periodic message triggering an instrument query for telemetry data.

Immediate Descendants of the measure routine should override the measure routine by implementing class specific override methods. The immediate descendant is the only class that should override measure, and their descendants should provide an application specefic method for acquiring data.

This approach guarantees a common reporting method for instrument telemetry regardless of the specific device used,

Version: 1.0.0.0

3.2.1. Diagram
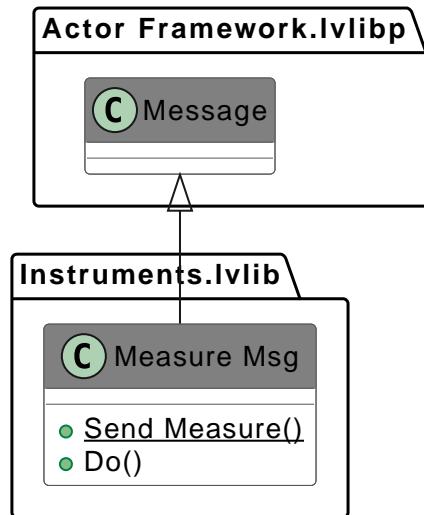
3.2.2. Methods

Table 3. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Send Measure		This VI sends the message to an actor.		▣	⊞
Do		This VI delivers the message to the actor by calling the appropriate method(s) on the actor.		⑤	

Scope: 🔑 → Protected | 🔑 → Community

Reentrancy: ▣ → Preallocated reentrancy | ⑤ → Shared reentrancy

Inlining: ⊞ → Inlined
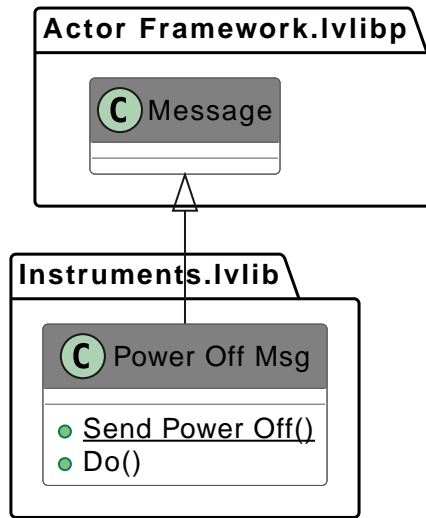
3.2.3. Class Constant VIs

NOTE No Constant VIs Found

3.3. Power Off Msg.lvclass

Responsibility: No description found (add content in lvclass description)

Version: 1.0.0.0

3.3.1. Diagram

3.3.2. Methods

Table 4. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Send Power Off		This VI sends the message to an actor.			
Do		This VI delivers the message to the actor by calling the appropriate method(s) on the actor.			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

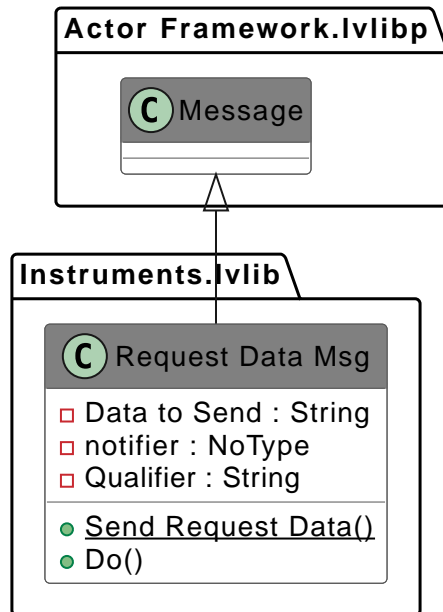**I**nlining: → Inlined

3.3.3. Class Constant VIs

NOTE   No Constant VIs Found

3.4. Request Data Msg.lvclass

Responsibility: No description found (add content in lvclass description)

Version: 1.0.0.0

3.4.1. Diagram

3.4.2. Methods

Table 5. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Send Request Data		This VI sends the message to an actor.		▣	⊞
Do		This VI delivers the message to the actor by calling the appropriate method(s) on the actor.		⑤	

Scope: 🗝 → Protected | 🗝 → Community

Reentrancy: ▣ → Preallocated reentrancy | ⑤ → Shared reentrancy

Inlining: ⊞ → Inlined

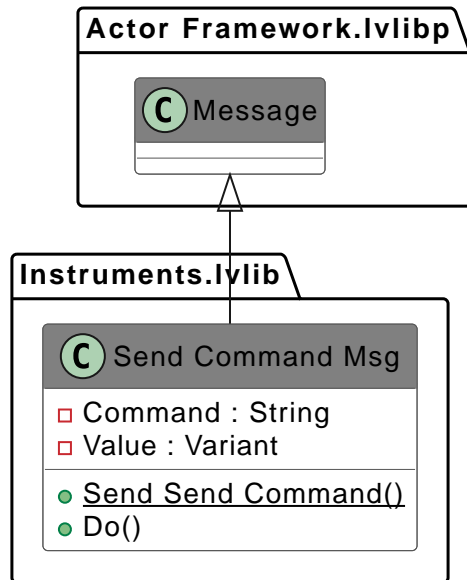### 3.4.3. Class Constant VIs

> **NOTE**   No Constant VIs Found

3.5. Send Command Msg.lvclass

Responsibility: No description found (add content in lvclass description)

Version: 1.0.0.0

3.5.1. Diagram

3.5.2. Methods

Table 6. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Send Send Command		This VI sends the message to an actor.			
Do		This VI delivers the message to the actor by calling the appropriate method(s) on the actor.			

Scope: ⚷ → Protected | ⚷ → Community

Reentrancy: 🅿 → Preallocated reentrancy | 🆂 → Shared reentrancy

Inlining: → Inlined
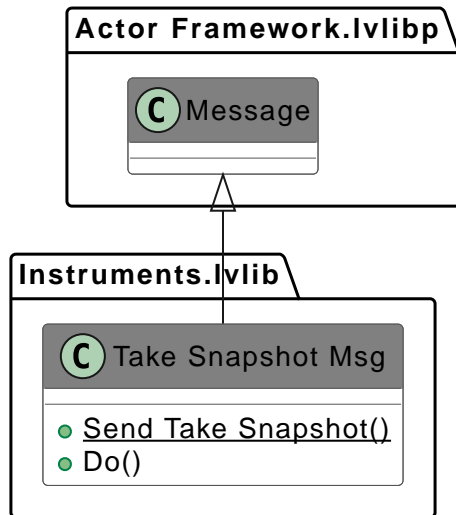
3.5.3. Class Constant VIs

NOTE No Constant VIs Found

3.6. Take Snapshot Msg.lvclass

Responsibility: No description found (add content in lvclass description)

Version: 1.0.0.0

3.6.1. Diagram

3.6.2. Methods

Table 7. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Send Take Snapshot		This VI sends the message to an actor.		▣	⊞
Do		This VI delivers the message to the actor by calling the appropriate method(s) on the actor.		⑤	

Scope: ⚷ → Protected | ⚷ → Community

Reentrancy: ▣ → Preallocated reentrancy | ⑤ → Shared reentrancy

Inlining: ⊞ → Inlined

3.6.3. Class Constant VIs

NOTE | No Constant VIs Found

Chapter 4. Actors (AF)

This section describes AF framework usage in the project

4.1. Preamble

Add anything that could be interseting to describe AF concepts and help the reader to understand the AF section
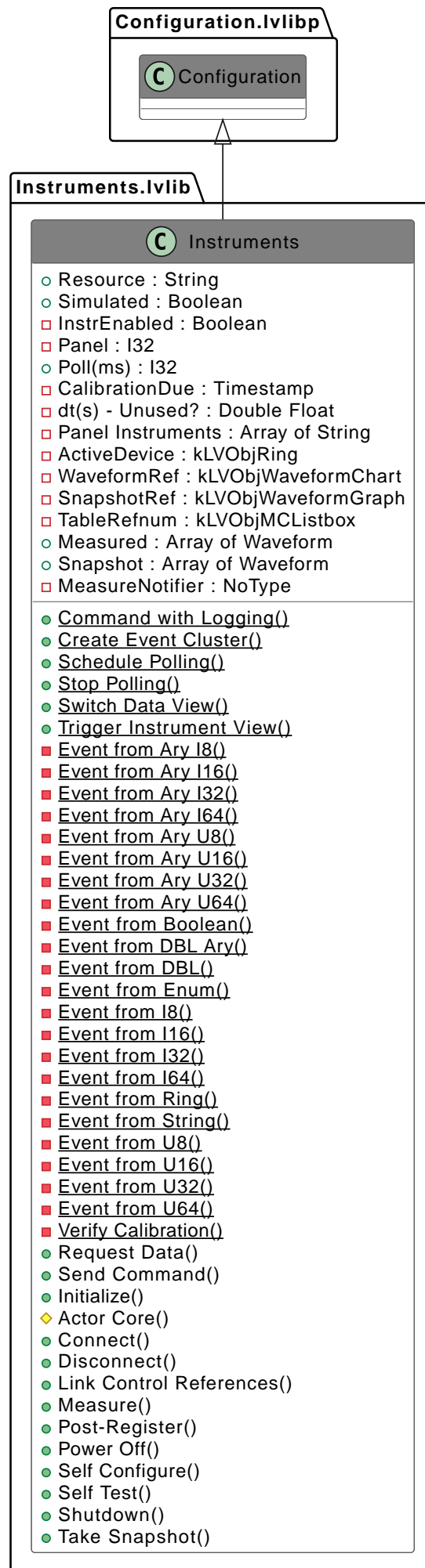
4.2. Actors overview

4.3. Instruments.lvclass

Responsibility: The **Instruments** Class is the abstraction layer for any instrument implemented in the software. It ensures that descendant classes implement the necessary code, and performs basic common operations that are universally used (reducing software development efforts)
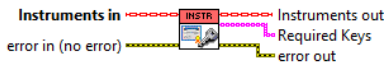
Version: 1.0.0.8

4.3.1. Diagram

Configuration.lvlibp

C Configuration

Instruments.lvlib

C Instruments

- ○ Resource : String
- ○ Simulated : Boolean
- □ InstrEnabled : Boolean
- □ Panel : I32
- ○ Poll(ms) : I32
- □ CalibrationDue : Timestamp
- □ dt(s) - Unused? : Double Float
- □ Panel Instruments : Array of String
- □ ActiveDevice : kLVObjRing
- □ WaveformRef : kLVObjWaveformChart
- □ SnapshotRef : kLVObjWaveformGraph
- □ TableRefnum : kLVObjMCListbox
- ○ Measured : Array of Waveform
- ○ Snapshot : Array of Waveform
- □ MeasureNotifier : NoType

- ● Command with Logging()
- ● Create Event Cluster()
- ● Schedule Polling()
- ● Stop Polling()
- ● Switch Data View()
- ● Trigger Instrument View()
- ■ Event from Ary I8()
- ■ Event from Ary I16()
- ■ Event from Ary I32()
- ■ Event from Ary I64()
- ■ Event from Ary U8()
- ■ Event from Ary U16()
- ■ Event from Ary U32()
- ■ Event from Ary U64()
- ■ Event from Boolean()
- ■ Event from DBL Ary()
- ■ Event from DBL()
- ■ Event from Enum()
- ■ Event from I8()
- ■ Event from I16()
- ■ Event from I32()
- ■ Event from I64()
- ■ Event from Ring()
- ■ Event from String()
- ■ Event from U8()
- ■ Event from U16()
- ■ Event from U32()
- ■ Event from U64()
- ■ Verify Calibration()
- ● Request Data()
- ● Send Command()
- ● Initialize()
- ◆ Actor Core()
- ● Connect()
- ● Disconnect()
- ● Link Control References()
- ● Measure()
- ● Post-Register()
- ● Power Off()
- ● Self Configure()
- ● Self Test()
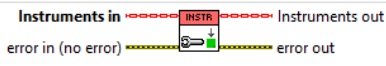- ● Shutdown()
- ● Take Snapshot()

4.3.2. Methods

Table 8. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Read IstrEnabled		No description found (add content in vi description)		▣	⊞
Read Panel index		No description found (add content in vi description)		▣	⊞
Read Poll(ms)		No description found (add content in vi description)		▣	⊞
Read Resource		No description found (add content in vi description)		▣	⊞
Read Simulated		No description found (add content in vi description)		▣	⊞
Read Snapshot		No description found (add content in vi description)		▣	⊞
Write Snapshot		No description found (add content in vi description)		▣	⊞
Read Measured		No description found (add content in vi description)		▣	⊞
Write Measured		No description found (add content in vi description)		▣	⊞
Command with Logging		No description found (add content in vi description)		▣	
Write TableData		No description found (add content in vi description)		▣	
Schedule Polling		Schedule periodic measure poll message at the defined poll rate This is called in the post-actor routine.			
Stop Polling		This stops the scheduled measurement polling			
Switch Data View		**Switch Data View** This method allows a developer to switch displays from waveform, table, and snapshot data. This is a method that's necessary for multiple instruments.			
Trigger Instrument View		**Trigger Instrument View** This generates a change instrument event, changing what instrument is displayed in the panel, allowing a station to support many instruments with few display panes.			

Name	Connector pane	Description	S.	R.	I.
Request Data		**Request Data** This is the automation hook to Request Data from any ancestor instrument: **Inputs** String **Data to Send** General Request string, defined by the ancestor classes Variant Noitifier**notifier** This is the notifier used to control sequencing by the automation tool (ie response data is pushed into the notifier when the data is ready) String **Qualifier** Specific request string (example may be request measured as data to send, and channel 1 as a qualifier, resulting in a return of only channel 1 measurement data)		🖫	
Send Command		**Send Command** This is the automation hook to command descendant instruments to perform simple functions.		🖫	
Initialize		This is an abstraction that is already implemented in the pre-launch init routine Initialize is called **after** the event reference queues and caller and self enqueuers are created. Be sure to inspect the parent method to determine whether or not call parent method is required.			
Actor Core		This actor core defines the GUI scaling for ancestor classes.	🗝	🖫	
Connect		**Connect** Descendents must override and then call this method. When called, this method will call the self-test routine and check for any self test errors.			
Disconnect		**Disconnect** Descendents must override and then call this method.			

Name	Connector pane	Description	S.	R.	I.
Link Control References	Instruments in — Control Refs — error in (no error) — INSTR — Instruments out / error out	This dynamically dispatched routine is meant to route any provided control references to class private references by name (label text)		▣	
Measure	Instruments in — error in (no error) — INSTR — Instruments out / error out	**Measure** Must be overridden by descendent classes. Call parent method must be called **after** the override **Instrument** layer reads the measured private data, and enqueues the data for logging Immediate Descendants of the measure routine should override the measure routine by implementing class specific override methods. The immediate descendant is the only class that should override measure, and their descendants should provide an application speceific method for acquiring data. This approach guarantees a common reporting method for instrument telemetry regardless of the specific device used.		▣	13
Post-Register	Instruments in — error in (no error) — INSTR — Instruments out / error out	**Post Register** This routine is implemented in **Post Launch Initialize** after the actor has registered for events. It's intended to provide a means of performing actions immediately before the actor state engine is entered.			
Power Off	Instruments in — error in (no error) — INSTR POWER OFF — Instruments out / error out	Override Class at the **Instrument** layer This is called in the **Shutdown** routine at the **Instrument** layer, so really only needs to be overidden at the base (device specific) class.			

Name	Connector pane	Description	S.	R.	I.
Read Required Keys		**Instruments** At this layer additional information is necessary to define instrument functionality universally **Keys Calibration** - Calibration Due Date (NCR = No Cal Required) **Resource** - String representation of the connection resource **Poll(ms)** - Measurement Query interval (how often should the GUI update with telemetry) **Panel** - Docking Position of the instrument in the manual GUI **Simulated** - True/False is this implementation an simulated instrument? **Enabled** - Enabled - This allows developers to integrate the test instruments in stages		⑤	
Read Required Signals		This override routine is used to create default signals to be loaded in to the signal list. This is a placeholder routine for editor functions in later development. **Instrument** - There is no implementation at the instrument layer, this is provided as a pass through.			
Self Configure		**Self Configure** This routine reads all of the required keys from **Required Configuration Keys** and updates the private data with the necessary information to set up the instrument. Then it tests the calibration information to determine if the instrument is within the cal cycle.			
Self Test		**Self Test** Descendents must override this routine Self test is meant to check the health of an instrument after the device has been connected. This routine is called in the Instrument Class Connect routine			

Name	Connector pane	Description	S.	R.	I.
Shutdown	Instruments in → Instruments out, error in (no error) → error out	**Shutdown** This routine is implemented in the stop actor override → as a result, it is not necessary to call shutdown anywhere else, and any methods that need to be called prior to stop actor core can be implemented here. **Engine** - No methods implemented at the engine level **Managed Actor** This routine transmits a stop core to all subordinate actors			
Take Snapshot	Instruments in → Instruments out, error in → error out	**Take Snapshot** This routine is for instruments that support triggered acquisition, override routines should configure and apply trigger sources, wait for the triggered event, and update the snapshot data record when the trigger is received, or throw a timeout warning if the triggered event didn't happen within the timeout range.			

Scope: ⚿ → Protected | ⚿ → Community

Reentrancy: 🄿 → Preallocated reentrancy | 🅂 → Shared reentrancy

Inlining: → Inlined

4.3.3. Class Constant VIs

NOTE | No Constant VIs Found

Chapter 5. Legal Information

5.1. Document creation

This document has been generated using the following tools.

5.1.1. Antidoc

Project website: Antidoc

Maintainer website: Wovalab

BSD 3-Clause License

Copyright © 2019-2025, Wovalab, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

5.1.2. Asciidoc for LabVIEW™

Project website: Asciidoc toolkit

Maintainer website: Wovalab

BSD 3-Clause License

Copyright © 2019-2025, Wovalab, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

5.1.3. classy Diagram Viewer

Project website: classy Diagram Viewer

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

5.2. Product used in the project

Antidoc hasn't been able to detect third party products in the project. This is the author's responsibility to list any of the missing product used.