Vending Machine

Antidoc v3.0.0, Enrique

Table of Contents

Chapter 1. Project description

No description found (add content in project description)

Chapter 2. DQMH® modules

This section describes DQMH® module responsibilities and relationships.

2.1. Preamble

A DQMH module is the main component of an architecture based on DQMH® framework. A DQMH module is used to implement a section of the application that has one responsibility.

DQMH® framework defines two different type of DQMH module.

Singleton:

A Singleton DQMH module can have only one instance running at any given time.

Cloneable:

A Cloneable DQMH module can have one or multiple instances running in parallel.

DQMH® framework defines two different ways to carry data throughout the application and with both other DQMH modules and non-DQMH based code.

Request events:

A request is a code that fires an event requesting the DQMH module to do something. Multiple locations in the code can send events to the DQMH module.

Request events are many-to-one.

Requests are usually named using imperative tense.

Broadcast events:

A broadcast is a code that fires an event broadcasting that the DQMH module did something. Multiple Event Structures can register to handle the Broadcast Events.

Broadcast Events are one-to-many.

Broadcasts are usually named using past tense or passive voice.

Default events:

By default all the DQMH ® Modules are shipped with some default events (Requests and Broadcasts). Here is the list of this events:

- Start Module
- Stop Module
- Show Panel
- Hide Panel
- Show Diagram
- Status Updated
- Error Reported
- Module Did Stop
- Update Module Execution Status
- Module Did Init
- Get Module Execution Status

If you want to learn more about default DQMH ® Events see the documentaiton.

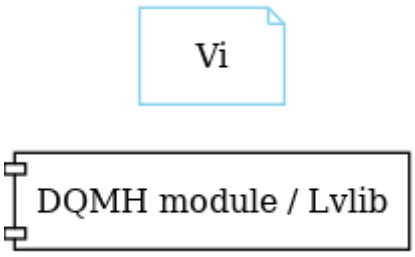
NOTE	Refer to the DQMH® framework official documentation to find more details on how the framework works

The following section gives you details on the project architecture relying on this framework. It gives you an overview of the modules' interaction and detailed information on each module.

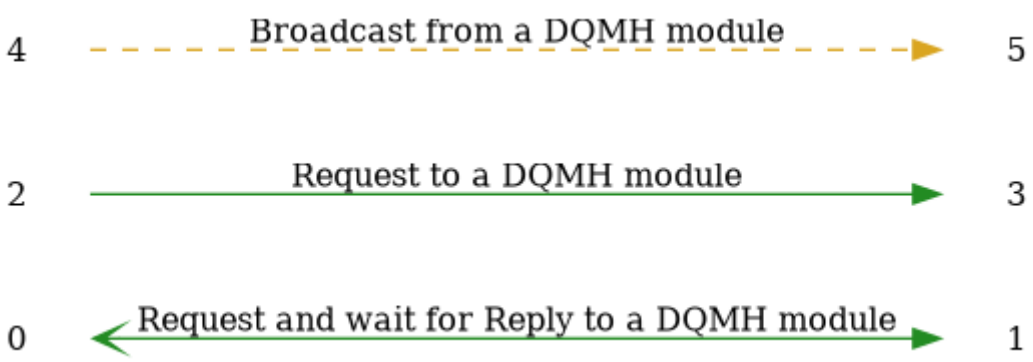
NOTE	If your module has a helper loop, it will be listed along with the events it processes. Helper loops can be categorized as either DQMH or Custom. To ensure your helper loop adheres to DQMH guidelines see the documentaiton.

Graphs used in this section have the following legend:

Components:

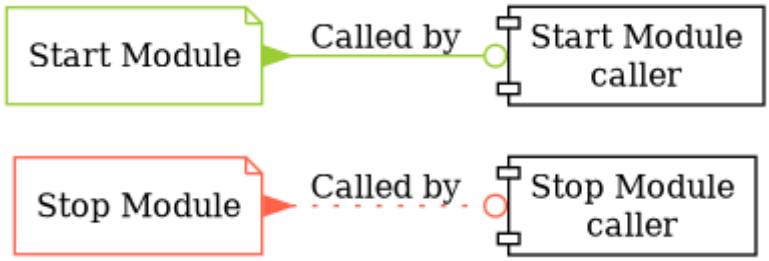


Events:



NOTE	One arrow can represent one or more events between two components
NOTE	Request and Request and wait for Reply are represented by only one arrow. If there is no Request and wait for Reply, Request representation is used. Otherwise Request and wait for Reply is used

Start and Stop module callers:



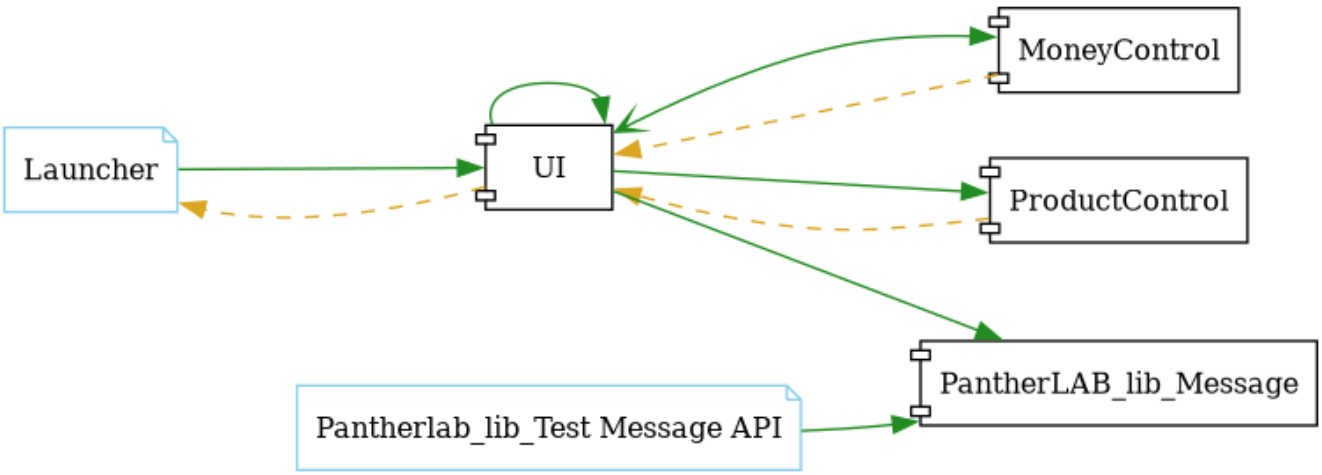
2.2. Modules overview

This project contains 3 singleton modules and 1 cloneable module.

Table 1. Modules list

Singleton	Cloneable
MoneyControl.lvlib	PantherLAB_lib_Message.lvlib
ProductControl.lvlib	
UI.lvlib	

This graph represents the links between all DQMH modules.



2.3. MoneyControl.lvlib

Type: Singleton

Responsibility: Modulo para el manejo de dinero, detecta el ingreso del dinero, ademas maneja el credito

2.3.1. Module relationship

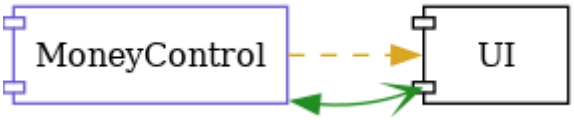


Table 2. Requests callers

Request Name	Callers
Devuelve dinero	
Draw FP in Subpanel	UI.lvlib:Main.vi
Entrega Cambio	UI.lvlib:Main.vi
Get Module Execution Status	
Hide Panel	
Incrementa Credito	
Remove From Subpanel	
Show Diagram	
Show Panel	
consulta credito	UI.lvlib:Main.vi

Table 3. Broadcasts Listeners

Broadcast Name	Listeners
Cambio Actualizado	UI.lvlib:Main.vi
Credito Actualizado	UI.lvlib:Main.vi

Broadcast Name	Listeners
Error Reported	UI.lvlib:Main.vi
Module Did Init	UI.lvlib:Main.vi
Module Did Stop	
Status Updated	UI.lvlib:Main.vi
Update Module Execution Status	

Table 4. Used requests

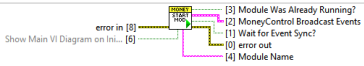
Module	Requests
—	—

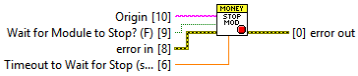
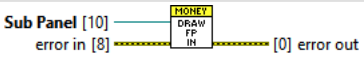
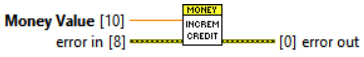
Table 5. Registered broadcast

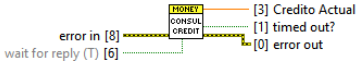
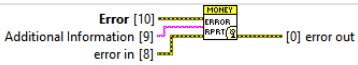
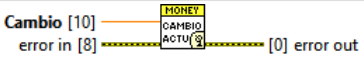
Module	Broadcasts
—	—

2.3.2. Event list

Table 6. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a **Register For Events** function. After the optional Register For Events function call, you should always call the **Synchronize Module Events.vi** for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Start Module: Value Change" event frame in the API Tester VI for this module. The **Validate DQMH Module** tool added a 'Module Name' output to this VI.			

Name	Type	Connector pane	Description	S.	R.	I.
Stop Module			Send the Stop request to the Module's Main.vi. If **Wait for Module to Stop?** is TRUE, this VI will wait until the module main VI stops, and will timeout at the **Timeout to Wait for Stop** value. This value defaults to "-1", which means the VI will not timeout, and will always wait until the module main VI stops before completing execution. Note: The **Timeout to Wait for Stop** value is ignored if 'Wait for Module to Stop?' is set to FALSE.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			
Get Module Execution Status			Fire the Get Module Execution Status request.			
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Draw FP in Subpanel			Draw FP in Subpanel			
Remove From Subpanel			Remove From Subpanel			
Incrementa Credito			Al detectar el ingreso de monedas y o billetes deberá actualizar el credito, utiliza el credito previo y suma la cantidad ingresada.			
Devuelve dinero			Esta accion es la de cancelar la compra			
Entrega Cambio			Request para entregar cambio despues de hacer venta			

Name	Type	Connector pane	Description	S.	R.	I.
consulta credito						
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Status Updated			Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported			Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop			Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status			Broadcast event to specify whether or not the module is running.			
Credito Actualizado			Este es el broadcast para actualizar el credito.			
Cambio Actualizado			Este es el broadcast para actualizar el Cambio.			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

2.3.3. Module Start/Stop calls

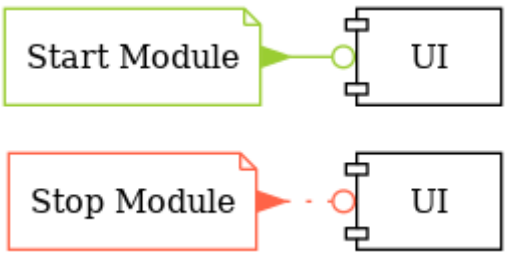


Table 7. Start and Stop module callers

Function	Callers
Start Module	UI.lvlib:Main.vi
Stop Module	UI.lvlib:Main.vi

2.3.4. Module Helper Loops

NOTE	No Helper Loops Found

2.3.5. Module custom errors

TIP	Custom errors are added to the module via vi named `*--error.vi`.

Module MoneyControl.lvlib use the following custom errors:

Table 8. Custom errors

Name	Code	Description
Module Not Running	0	Error information not found in the code
Module Not Stopped	403682	The Stop Module VI for the %s module timed out while waiting for the module main VI to stop. The module main VI may still be running.
Module Not Synced	403683	%s Module was unable to synchronize events.
Request and Wait for Reply Timeout	403686	

2.3.6. Module Constant VIs

Table 9. Constant VIs Found

VI Name	Data Type	Value
Module Name—constant.vi	String	MoneyControl
Module Timeout—constant.vi	I32	5000

2.4. ProductControl.lvlib

Type: Singleton

Responsibility: No description found (add content in DQMH module lvlib description)

2.4.1. Module relationship

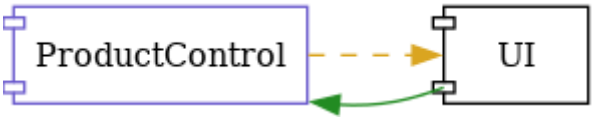


Table 10. Requests callers

Request Name	Callers
Get Module Execution Status	
Hide Panel	
Show Diagram	
Show Panel	
Vender Producto	UI.lvlib:Main.vi

Table 11. Broadcasts Listeners

Broadcast Name	Listeners
Error Reported	
Inventario Actualizado	UI.lvlib:Main.vi
Module Did Init	
Module Did Stop	
Status Updated	
Update Module Execution Status	

Table 12. Used requests

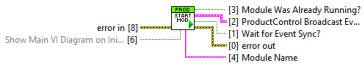
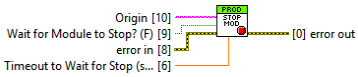
Module	Requests
—	—

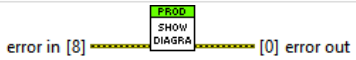
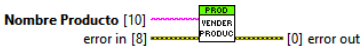
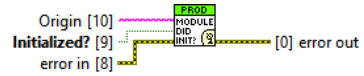
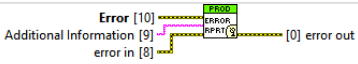
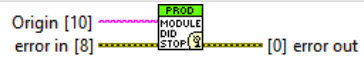
Table 13. Registered broadcast

Module	Broadcasts
—	—

2.4.2. Event list

Table 14. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a **Register For Events** function. After the optional Register For Events function call, you should always call the **Synchronize Module Events.vi** for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Start Module: Value Change" event frame in the API Tester VI for this module. The **Validate DQMH Module** tool added a 'Module Name' output to this VI.			
Stop Module			Send the Stop request to the Module's Main.vi. If **Wait for Module to Stop?** is TRUE, this VI will wait until the module main VI stops, and will timeout at the **Timeout to Wait for Stop** value. This value defaults to "-1", which means the VI will not timeout, and will always wait until the module main VI stops before completing execution. Note: The **Timeout to Wait for Stop** value is ignored if 'Wait for Module to Stop?' is set to FALSE.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			

Name	Type	Connector pane	Description	S.	R.	I.
Get Module Execution Status			Fire the Get Module Execution Status request.			
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Vender Producto						
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Status Updated			Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported			Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop			Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status			Broadcast event to specify whether or not the module is running.			
Inventario Actualizado			Broadcast para refrescar el inventario cada vez que se modifique			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

2.4.3. Module Start/Stop calls

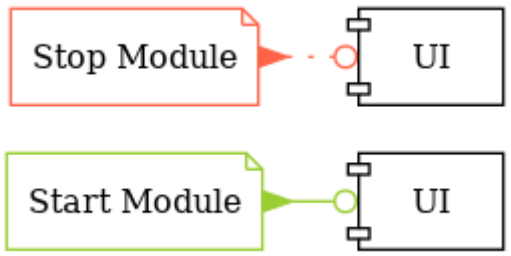


Table 15. Start and Stop module callers

Function	Callers
Start Module	UI.lvlib:Main.vi
Stop Module	UI.lvlib:Main.vi

2.4.4. Module Helper Loops

NOTE No Helper Loops Found

2.4.5. Module custom errors

TIP Custom errors are added to the module via vi named `*--error.vi`.

Module ProductControl.lvlib use the following custom errors:

Table 16. Custom errors

Name	Code	Description
Module Not Running	0	Error information not found in the code
Module Not Stopped	403682	The Stop Module VI for the %s module timed out while waiting for the module main VI to stop. The module main VI may still be running.
Module Not Synced	403683	%s Module was unable to synchronize events.
Request and Wait for Reply Timeout	403686	

2.4.6. Module Constant VIs

Table 17. Constant VIs Found

VI Name	Data Type	Value
Inventario—constant.vi	["Path","Path"]	["Inventario.ini","data\\Inventario.ini"]
Module Name—constant.vi	String	ProductControl
Module Timeout—constant.vi	I32	5000

2.5. UI.lvlib

Type: Singleton

Responsibility: Modulo principal

2.5.1. Module relationship

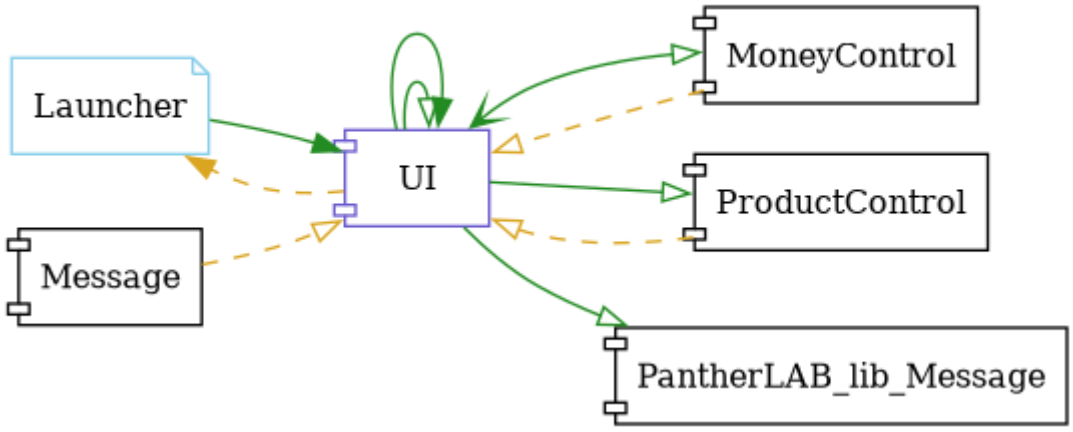


Table 18. Requests callers

Request Name	Callers
Get Module Execution Status	
Hide Panel	
Lanza otros modulos	UI.lvlib:Main.vi
Show Diagram	
Show Panel	Launcher.vi

Table 19. Broadcasts Listeners

Broadcast Name	Listeners
Error Reported	Launcher.vi
Module Did Init	Launcher.vi
Module Did Stop	Launcher.vi
Status Updated	Launcher.vi
Update Module Execution Status	Launcher.vi

Table 20. Used requests

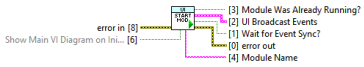
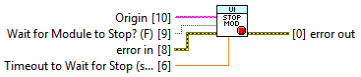
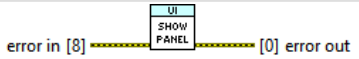
Module	Requests
MoneyControl.lvlib	Draw FP in Subpanel.vi (2) Entrega Cambio.vi Get Module Execution Status.vi (2) Stop Module.vi (2) consulta credito.vi
PantherLAB_lib_Message.lvlib	Display Message.vi (5) Draw FP in Subpanel.vi Stop Module.vi (2)
ProductControl.lvlib	Get Module Execution Status.vi Stop Module.vi (2) Vender Producto.vi
UI.lvlib	Lanza otros modulos.vi

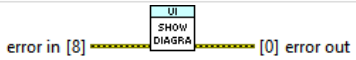
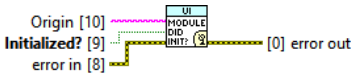
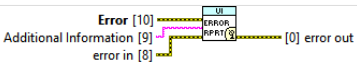
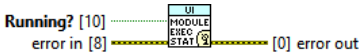
Table 21. Registered broadcast

Module	Broadcasts
Message.lvlib	Error Reported.vi Module Did Init.vi Status Updated.vi
MoneyControl.lvlib	Cambio Actualizado.vi Credito Actualizado.vi Error Reported.vi Module Did Init.vi Status Updated.vi
ProductControl.lvlib	Inventario Actualizado.vi

2.5.2. Event list

Table 22. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a **Register For Events** function. After the optional Register For Events function call, you should always call the **Synchronize Module Events.vi** for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Start Module: Value Change" event frame in the API Tester VI for this module. The **Validate DQMH Module** tool added a 'Module Name' output to this VI.			
Stop Module			Send the Stop request to the Module's Main.vi. If **Wait for Module to Stop?** is TRUE, this VI will wait until the module main VI stops, and will timeout at the **Timeout to Wait for Stop** value. This value defaults to "-1", which means the VI will not timeout, and will always wait until the module main VI stops before completing execution. Note: The **Timeout to Wait for Stop** value is ignored if 'Wait for Module to Stop?' is set to FALSE.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			

Name	Type	Connector pane	Description	S.	R.	I.
Get Module Execution Status			Fire the Get Module Execution Status request.			
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Status Updated			Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported			Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop			Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status			Broadcast event to specify whether or not the module is running.			
Lanza otros modulos						

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

2.5.3. Module Start/Stop calls

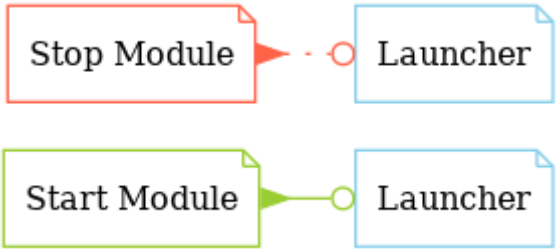


Table 23. Start and Stop module callers

Function	Callers
Start Module	Launcher.vi
Stop Module	Launcher.vi

2.5.4. Module Helper Loops

NOTE No Helper Loops Found

2.5.5. Module custom errors

TIP Custom errors are added to the module via vi named `*--error.vi`.

Module UI.lvlib use the following custom errors:

Table 24. Custom errors

Name	Code	Description
Module Not Running	0	Error information not found in the code
Module Not Stopped	403682	The Stop Module VI for the %s module timed out while waiting for the module main VI to stop. The module main VI may still be running.
Module Not Synced	403683	%s Module was unable to synchronize events.
Request and Wait for Reply Timeout	403686	

2.5.6. Module Constant VIs

Table 25. Constant VIs Found

VI Name	Data Type	Value
Module Name—constant.vi	String	UI
Module Timeout—constant.vi	I32	5000

2.6. PantherLAB_lib_Message.lvlib

Type: Cloneable

Responsibility: No description found (add content in DQMH module lvlib description)

2.6.1. Module relationship

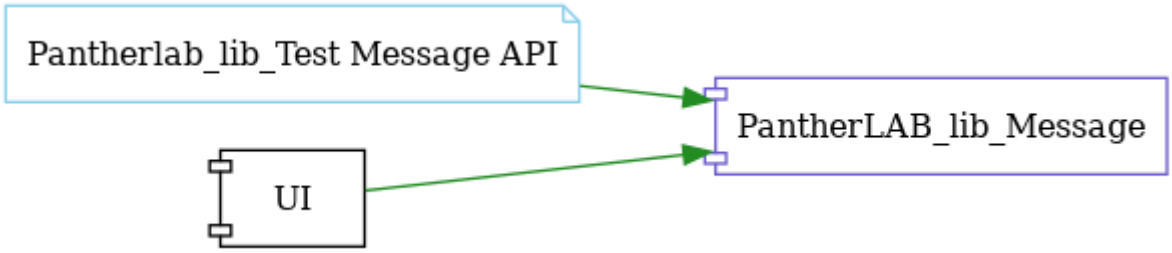


Table 26. Requests callers

Request Name	Callers
Display Message	Pantherlab_lib_Test Message API.vi UI.lvlib:Main.vi
Draw FP in Subpanel	Pantherlab_lib_Test Message API.vi UI.lvlib:Main.vi
Hide Panel	Pantherlab_lib_Test Message API.vi
Remove from Subpanel	Pantherlab_lib_Test Message API.vi
Set Blinking	Pantherlab_lib_Test Message API.vi
Set Display Style	Pantherlab_lib_Test Message API.vi
Set Text Colors	Pantherlab_lib_Test Message API.vi
Set Text Font Settings	Pantherlab_lib_Test Message API.vi
Show Diagram	Pantherlab_lib_Test Message API.vi
Show Panel	Pantherlab_lib_Test Message API.vi

Table 27. Broadcasts Listeners

Broadcast Name	Listeners
Error Reported	
Module Did Init	
Module Did Stop	
Status Updated	
Update Module Execution Status	

Table 28. Used requests

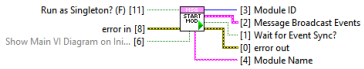
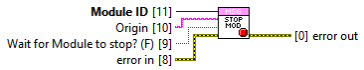
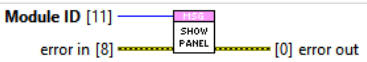
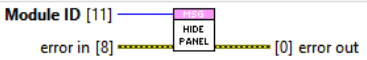
Module	Requests
—	—

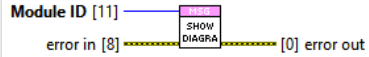
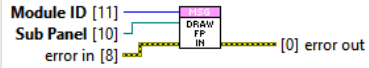
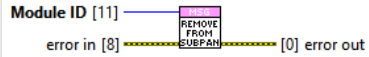
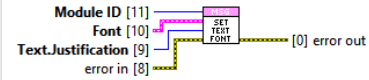
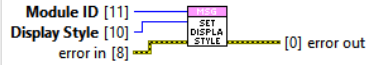
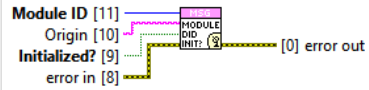
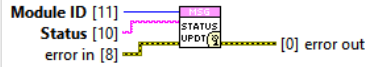
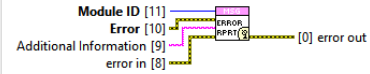
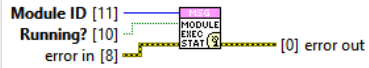
Table 29. Registered broadcast

Module	Broadcasts
—	—

2.6.2. Event list

Table 30. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches an instance of the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a **Register For Events** function. After the optional Register For Events function call, you should always call the **Synchronize Module Events.vi** for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Run New Module: Value Change" event frame in the API Tester VI for this module. The **Validate DQMH Module** tool added a 'Module Name' output to this VI.			
Stop Module			Send the Stop request to the Module's Main.vi. If **Wait for Module to stop?** is TRUE, then this VI will not complete execution until the Module Main VI has stopped running. **Note:** If the cloneable module is running as singleton, then the 'Wait for Module to stop?' input is ignored… this VI will **always** wait until a cloneable Main VI running as singleton has stopped running.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			

Name	Type	Connector pane	Description	S.	R.	I.
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Display Message			Display Message			
Draw FP in Subpanel			Draw FP in Subpanel			
Remove from Subpanel			Remove from Subpanel			
Set Text Font Settings			Set Text Font Settings			
Set Text Colors			Set Text Colors			
Set Blinking			Set Blinking Status			
Set Display Style			Set Display Style			
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Status Updated			Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported			Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop			Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status			Fire the Get Module Execution Status request.			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

2.6.3. Module Start/Stop calls

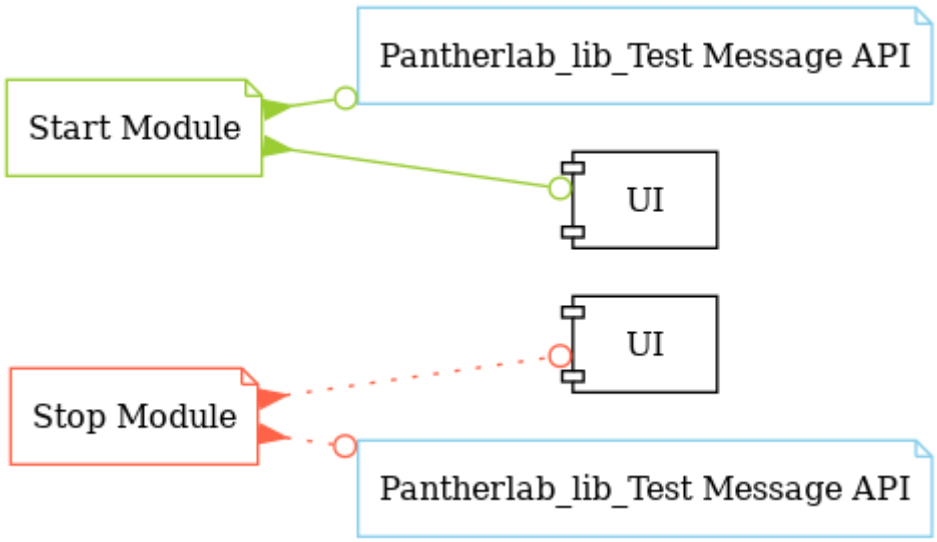


Table 31. Start and Stop module callers

Function	Callers
Start Module	UI.lvlib:Main.vi Pantherlab_lib_Test Message API.vi
Stop Module	UI.lvlib:Main.vi Pantherlab_lib_Test Message API.vi

2.6.4. Module Helper Loops

NOTE No Helper Loops Found

2.6.5. Module custom errors

TIP Custom errors are added to the module via vi named `*--error.vi`.

Module PantherLAB_lib_Message.lvlib use the following custom errors:

Table 32. Custom errors

Name	Code	Description
Module Running as Singleton	403680	The "%s" module is currently running as singleton, but the Start Module VI was called with 'Run as Singleton' specified as FALSE.
Module Not Stopped	403682	The Stop Module VI for the %s module timed out while waiting for the module main VI to stop. The module main VI may still be running.

Name	Code	Description
Module Not Synced	403683	%s Module was unable to synchronize events.
Module Not Running	403684	Not a single instance of "%s" Module running.
Module Running as Cloneable	403685	The "%s" module is currently running as cloneable, but the Start Module VI was called with 'Run as Singleton' specified as TRUE.
Request and Wait for Reply Timeout	403686	
Master Reference Not Closed	403687	The "%s" module cannot be run as singleton because the Master Reference is still open from a prior run as cloneable. If you plan on running this module as both singleton and cloneable, consider changing your Main VI to wire a value of TRUE to the 'Close Master Reference' input of Init Module.vi.

2.6.6. Module Constant VIs

Table 33. Constant VIs Found

VI Name	Data Type	Value
Module Name—constant.vi	["String","String"]	[" %d","Message"]
Module Timeout—constant.vi	I32	5000

Chapter 3. Libraries

This section describes the libraries contained in the project.

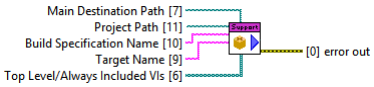
3.1. Launcher Support.lvlib

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

Version: 1.0.0.0

3.1.1. Functions

Table 34. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Determine if Running in Debug Mode		The launcher VI is meant to be run as a headless launcher for the module main VI. The launcher VI can be used as a debugging tool. This VI determines if the launcher VI is running as a debugger by parsing the command line arguments or checking if the VI is running in development mode and sets the Launcher VI properties accordingly.			
Pre-Build Action		This Pre-Build Action VI sets the Debug mode to false to ensure the top level VI has the headless properties at build time.			
Set VI Properties for Debugging Mode		This VI sets the properties of the VI Launcher to either headless or debugging mode.			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

3.1.2. Library Constant VIs

NOTE	No Constant VIs Found

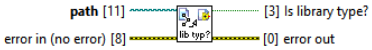
3.2. NI_FileType.lvlib

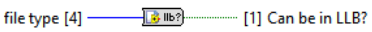
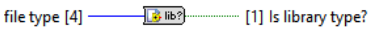
Responsibility: No description found (add content in lvlib description)

Version: 1.0.0.0

3.2.1. Functions

Table 35. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Can File be in LLB		Returns TRUE if this file could be copied into an LLB.			
Get File Type		This VI returns the file type of the file specified in "path". It does not load LabVIEW types (ex. VIs) into memory to determine this information. This VI works for LabVIEW types inside of LLBs. NOTE: This VI will not run in a built executable. An error 53 is returned.			
Get File Type Icon Image		Returns TRUE if this file could be loaded with the Open VI Reference function. It includes VIs, Polymorphic VIs, Controls, Templates, and Globals.			
Is File a LabVIEW document		Returns TRUE if this file could be opened by LabVIEW. It uses the same criteria as the File >> Open dialog.			
Is File a type of library		Returns TRUE if this file is a type of library, such as Libraries, XControls, XNodes, and LVClasses.			
Is File VI		Returns TRUE if this file could be loaded with the Open VI Reference function. It includes VIs, Polymorphic VIs, Controls, Templates, and Globals.			
Is File an LLB		Returns TRUE if this file is an LLB.			
Convert filetype to Is VI		Returns TRUE if this file type could be loaded with the Open VI Reference function. It includes VIs, Polymorphic VIs, Controls, Templates, and Globals.			
Convert filetype to Icon Image		Returns TRUE if this file type could be loaded with the Open VI Reference function. It includes VIs, Polymorphic VIs, Controls, Templates, and Globals.			

Name	Connector pane	Description	S.	R.	I.
Convert filetype to Can be in LLB		Returns TRUE if this file type could be copied into an LLB.			
Convert filetype to Is library type		Returns TRUE if this file type is a type of library, such as Libraries, XControls, XNodes, and LVClasses.			
Convert filetype to Is LabVIEW document		Returns TRUE if this file type could be opened by LabVIEW. It uses the same criteria as the File >> Open dialog.			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

3.2.2. Library Constant VIs

NOTE	No Constant VIs Found

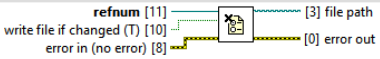
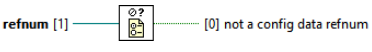
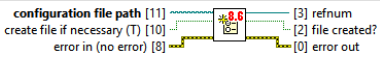
3.3. NI_LVConfig.lvlib

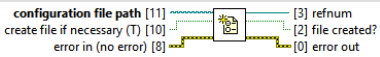
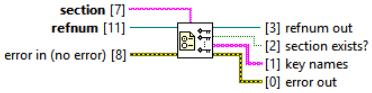
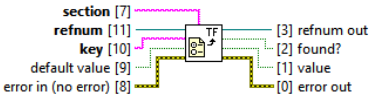
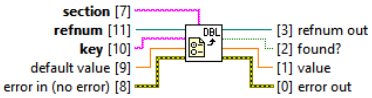
Responsibility: No description found (add content in lvlib description)

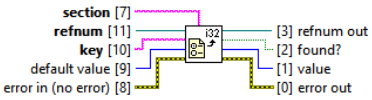
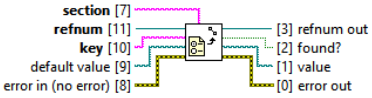
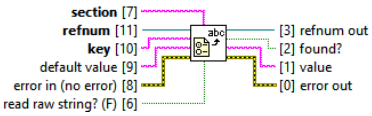
Version: 1.0.0.0

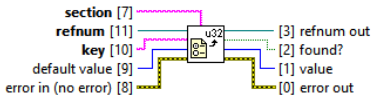
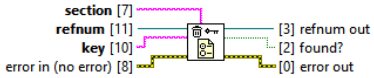
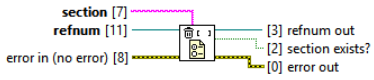
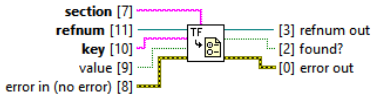
3.3.1. Functions

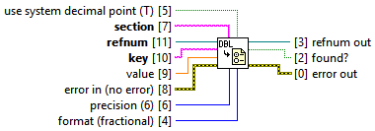
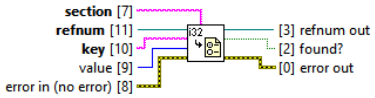
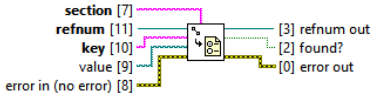
Table 36. Functions (non private scope only)

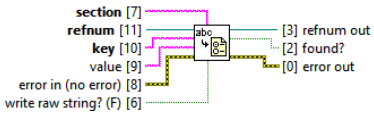
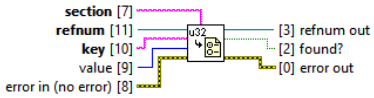
Name	Connector pane	Description	S.	R.	I.
Close Config Data		Writes data to the platform-independent configuration file identified by refnum and then closes the reference to that file.			
Not A Config Data Refnum		Determines whether a configuration data refnum is valid.			
Open Config Data (compatibility)		The Open Config Data VI was rewritten with one new output in LabVIEW 2009 and later. Replace the Open Config Data (compatibility) VI with an Open Config Data VI from the Functions palette to use the new functionality.			

Name	Connector pane	Description	S.	R.	I.
Open Config Data		Opens a reference to the configuration data found in a platform-independent configuration file.			
Get Key Names		Gets the names of all keys in the specified section from the configuration data identified by refnum.			
Get Section Names		Gets the names of all sections from the configuration data identified by refnum.			
Read Key (Boolean)		Reads a value associated with a key in a specified section from the configuration data identified by refnum. If the key does not exist, the VI returns the default value. This VI supports multibyte characters in strings. Wire data to the default value input to determine the polymorphic instance to use or manually select the instance.			
Read Key (Double)		Reads a value associated with a key in a specified section from the configuration data identified by refnum. If the key does not exist, the VI returns the default value. This VI supports multibyte characters in strings. Wire data to the default value input to determine the polymorphic instance to use or manually select the instance.			

Name	Connector pane	Description	S.	R.	I.
Read Key (I32)		Reads a value associated with a key in a specified section from the configuration data identified by refnum. If the key does not exist, the VI returns the default value. This VI supports multibyte characters in strings. Wire data to the default value input to determine the polymorphic instance to use or manually select the instance.			
Read Key (Path)		Reads a value associated with a key in a specified section from the configuration data identified by refnum. If the key does not exist, the VI returns the default value. This VI supports multibyte characters in strings. Wire data to the default value input to determine the polymorphic instance to use or manually select the instance.			
Read Key (String)		Reads a value associated with a key in a specified section from the configuration data identified by refnum. If the key does not exist, the VI returns the default value. This VI supports multibyte characters in strings. Wire data to the default value input to determine the polymorphic instance to use or manually select the instance.			

Name	Connector pane	Description	S.	R.	I.
Read Key (U32)		Reads a value associated with a key in a specified section from the configuration data identified by refnum. If the key does not exist, the VI returns the default value. This VI supports multibyte characters in strings. Wire data to the default value input to determine the polymorphic instance to use or manually select the instance.			
Remove Key		Removes a key in a specified section from the configuration data identified by refnum.			
Remove Section		Removes a section from the configuration data identified by refnum.			
Write Key (Boolean)		Writes a value to a key in a specified section of the configuration data identified by refnum. This VI modifies data in memory. To write data to disk, use the Close Config Data VI. Wire data to the value input to determine the polymorphic instance to use or manually select the instance.			

Name	Connector pane	Description	S.	R.	I.
Write Key (Double)		Writes a value to a key in a specified section of the configuration data identified by refnum. This VI modifies data in memory. To write data to disk, use the Close Config Data VI. Wire data to the value input to determine the polymorphic instance to use or manually select the instance.			
Write Key (I32)		Writes a value to a key in a specified section of the configuration data identified by refnum. This VI modifies data in memory. To write data to disk, use the Close Config Data VI. Wire data to the value input to determine the polymorphic instance to use or manually select the instance.			
Write Key (Path)		Writes a value to a key in a specified section of the configuration data identified by refnum. This VI modifies data in memory. To write data to disk, use the Close Config Data VI. Wire data to the value input to determine the polymorphic instance to use or manually select the instance.			

Name	Connector pane	Description	S.	R.	I.
Write Key (String)		Writes a value to a key in a specified section of the configuration data identified by refnum. This VI modifies data in memory. To write data to disk, use the Close Config Data VI. Wire data to the value input to determine the polymorphic instance to use or manually select the instance.			
Write Key (U32)		Writes a value to a key in a specified section of the configuration data identified by refnum. This VI modifies data in memory. To write data to disk, use the Close Config Data VI. Wire data to the value input to determine the polymorphic instance to use or manually select the instance.			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

3.3.2. Library Constant VIs

NOTE	No Constant VIs Found

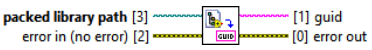
3.4. NI_PackedLibraryUtility.lvlib

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

Version: 1.0.0.0

3.4.1. Functions

Table 37. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Enable Caching		Use this VI to enable or disable packed library caching.			
Get Exported File List		Returns the qualified names and paths of the exported files in a packed project library.			
Get Exported File Path		Returns the path of an exported file in a packed project library.			
Get Guid String		Returns the path to the LabVIEW project (.lvproj) file this packed library was built from.			
Get Source Project Path		Returns the path to the LabVIEW project (.lvproj) file this packed library was built from.			
Packed Library Path		Returns the path to a packed project library for the VI you specify in vi reference.			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

3.4.2. Library Constant VIs

NOTE	No Constant VIs Found

Chapter 4. Classes

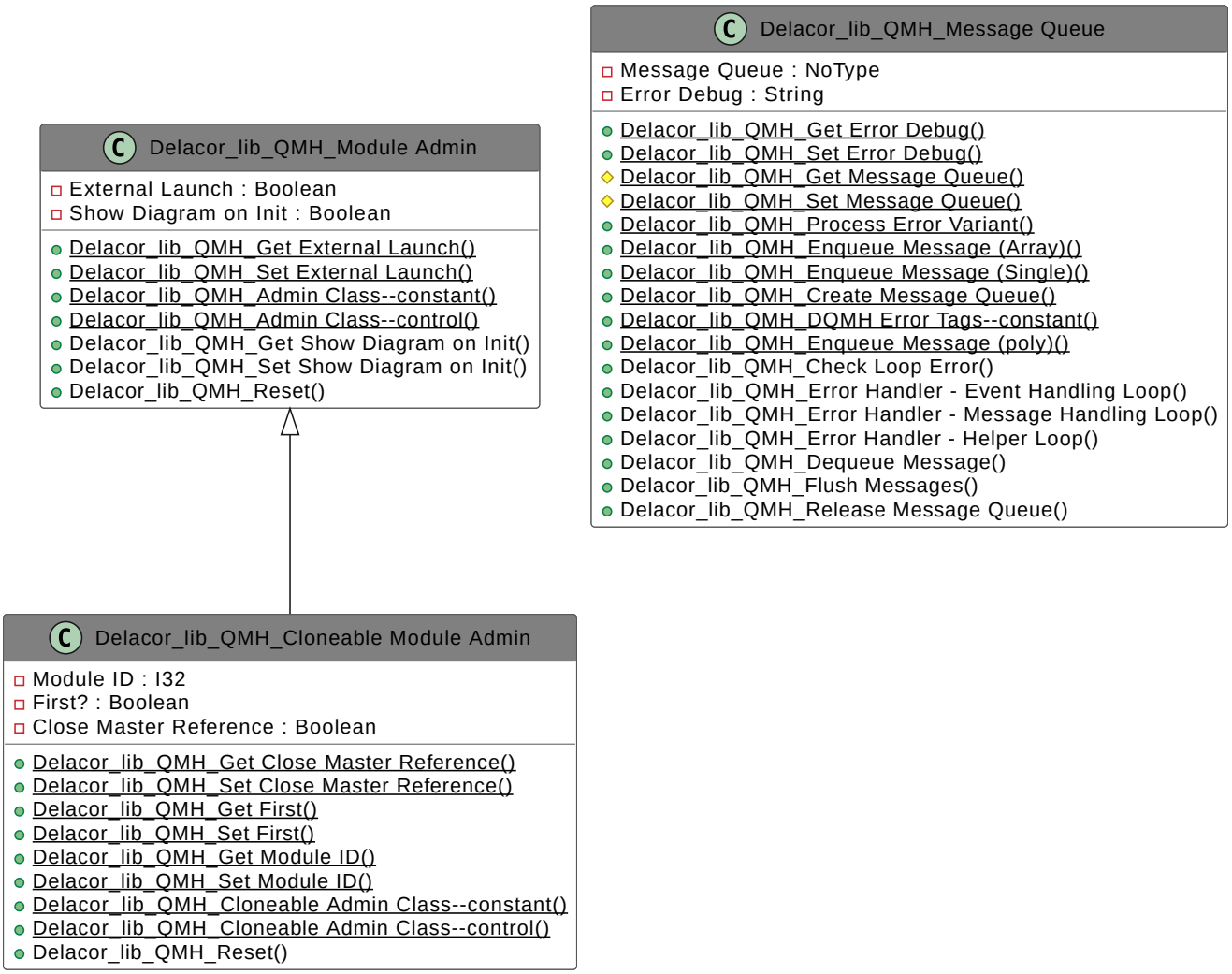
This section describes the classes contained in the project.

4.1. Classes overview

This project contains 3 classes and 0 interface.

Table 38. Classes list

Classes	Interfaces
Delacor_lib_QMH_Cloneable Module Admin.lvclass	
Delacor_lib_QMH_Message Queue.lvclass	
Delacor_lib_QMH_Module Admin.lvclass	

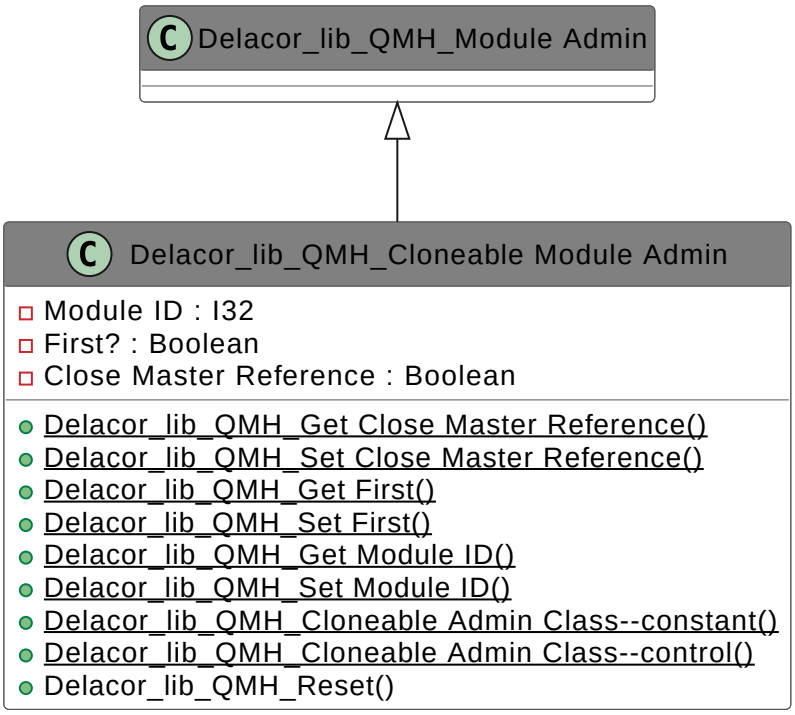


4.2. Delacor_lib_QMH_Cloneable Module Admin.lvclass

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

Version: 1.0.0.1

4.2.1. Diagram



4.2.2. Methods

Table 39. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Delacor_lib_QMH_Get Close Master Reference		Specifies whether or not the master VI reference used for launching clones should be closed by the Close Module VI when the cloneable module is shutting down. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Set Close Master Reference		Specifies whether or not the master VI reference used for launching clones should be closed by the Close Module VI when the cloneable module is shutting down. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Get First		Specifies whether or not this clone is the first one that was launched. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			

Name	Connector pane	Description	S.	R.	I.
Delacor_lib_QMH_Set First		Specifies whether or not this clone is the first one that was launched. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Get Module ID		The numeric identifier of a running instance of a cloneable module. If the module is running as a singleton, the value will be 0. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Set Module ID		The numeric identifier of a running instance of a cloneable module. If the module is running as a singleton, the value will be 0. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Cloneable Admin Class—control		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Reset		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

4.2.3. Class Constant VIs

Table 40. Constant VIs Found

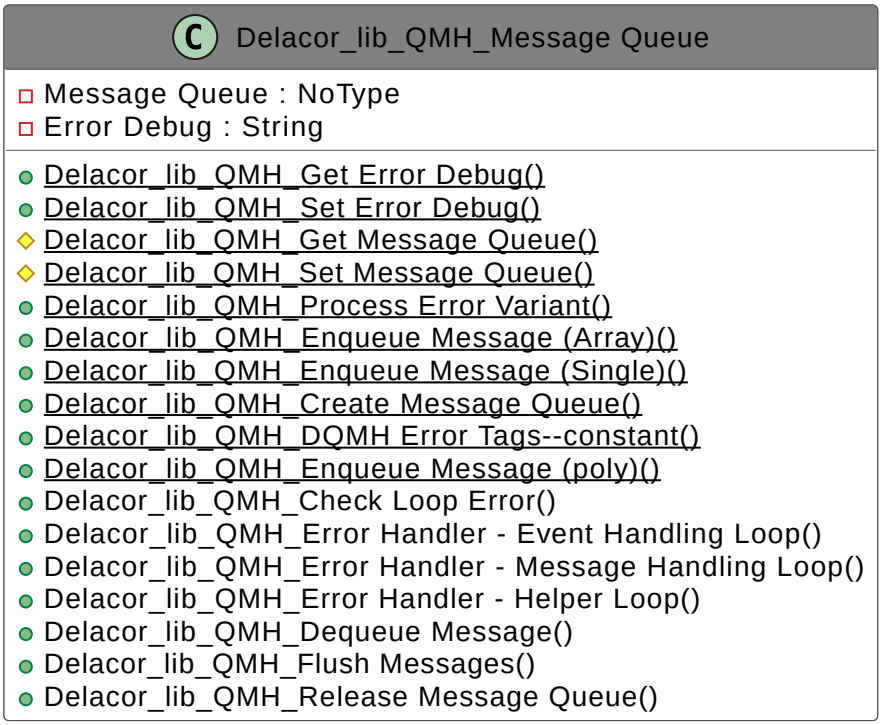
VI Name	Data Type	Value
Delacor_lib_QMH_Cloneable Admin Class—constant.vi	LabVIEW Class Instance	Delacor_lib_QMH_Cloneable Module Admin.lvclass

4.3. Delacor_lib_QMH_Message Queue.lvclass

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

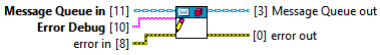
Version: 1.0.0.0

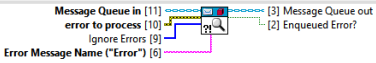
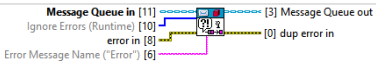
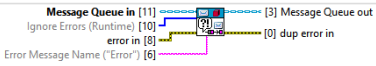
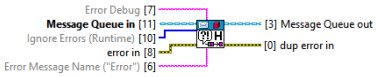
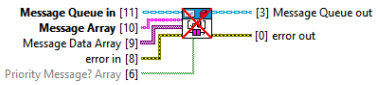
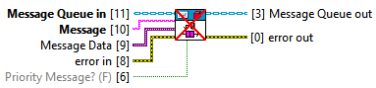
4.3.1. Diagram

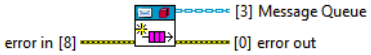
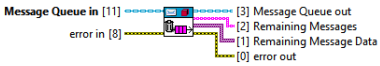


4.3.2. Methods

Table 41. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Delacor_lib_QMH_Get Error Debug		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Set Error Debug		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Get Message Queue		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Set Message Queue		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			

Name	Connector pane	Description	S.	R.	I.
Delacor_lib_QMH_Check Loop Error		Check the 'error to process' to see if its code value matches any of the values in the 'Ignore Errors' array. If so, do nothing. If not, send an "Error" message containing the error data to the Message Handling Loop for further processing. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Error Handler - Event Handling Loop		Process an error that occurred in the Event Handling Loop, either by ignoring it, or generating an "Error" message. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Error Handler - Message Handling Loop		Process an error that occurred in the Message Handling Loop, either by ignoring it, or generating an "Error" message. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Error Handler - Helper Loop		Process an error that occurred in a Helper Loop, either by ignoring it, or generating an "Error" message. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Process Error Variant		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Enqueue Message (Array)		For a regular message, enqueue the message. For a priority message, enqueue it at the front of the queue. This is an instance of the polymorphic VI: "Enqueue Message (poly).vi" __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Enqueue Message (Single)		For a regular message, enqueue the message. For a priority message, enqueue it at the front of the queue. This is an instance of the polymorphic VI: "Enqueue Message (poly).vi" __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			

Name	Connector pane	Description	S.	R.	I.
Delacor_lib_QMH_Create Message Queue		This VI creates and initializes the message queue for a QMH Module. If the message needs to be different for the given module, then create a child class of Message Queue and override the appropriate methods. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Dequeue Message		This VI pulls messages off the Message Queue. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Flush Messages		Flush the message queue. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Release Message Queue		Release the message queue. __ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

4.3.3. Class Constant VIs

Table 42. Constant VIs Found

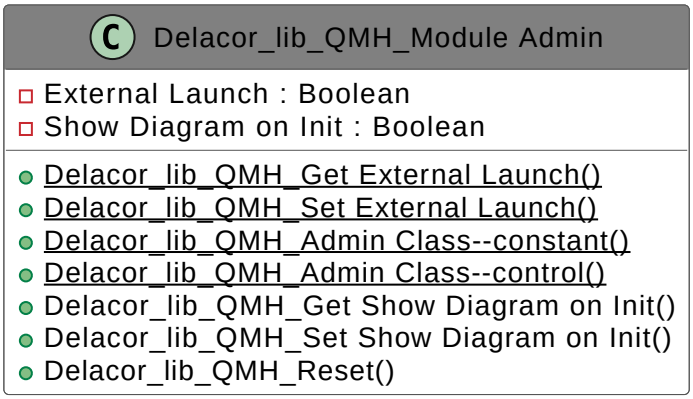
VI Name	Data Type	Value
Delacor_lib_QMH_DQMH Error Tags—constant.vi	["String","String"]	["<DQMH>","</DQMH>"]

4.4. Delacor_lib_QMH_Module Admin.lvclass

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

Version: 1.0.0.0

4.4.1. Diagram



4.4.2. Methods

Table 43. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Delacor_lib_QMH_Get External Launch		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Set External Launch		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Get Show Diagram on Init		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Set Show Diagram on Init		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Admin Class—control		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			
Delacor_lib_QMH_Reset		__ DQMH Framework: Palette 7.1.0.1503 Copyright (c) 2025, DQMH Consortium			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

4.4.3. Class Constant VIs

Table 44. Constant VIs Found

VI Name	Data Type	Value
Delacor_lib_QMH_Admin Class—constant.vi	LabVIEW Class Instance	Delacor_lib_QMH_Module Admin.lvclass

Chapter 5. Custom errors

TIP Custom errors are added via vi named `*--error.vi`.

Table 45. Custom errors

Name	Code	Description	Owned by
Module Not Running	0		MoneyControl.lvlib ProductControl.lvlib UI.lvlib
Module Running as Singleton	403680	The "%s" module is currently running as singleton, but the Start Module VI was called with 'Run as Singleton' specified as FALSE.	PantherLAB_lib_Message.lvlib
Module Not Stopped	403682	The Stop Module VI for the %s module timed out while waiting for the module main VI to stop. The module main VI may still be running.	MoneyControl.lvlib ProductControl.lvlib UI.lvlib PantherLAB_lib_Message.lvlib
Module Not Synced	403683	%s Module was unable to synchronize events.	MoneyControl.lvlib ProductControl.lvlib UI.lvlib PantherLAB_lib_Message.lvlib
Module Not Running	403684	Not a single instance of "%s" Module running.	PantherLAB_lib_Message.lvlib
Module Running as Cloneable	403685	The "%s" module is currently running as cloneable, but the Start Module VI was called with 'Run as Singleton' specified as TRUE.	PantherLAB_lib_Message.lvlib
Request and Wait for Reply Timeout	403686		MoneyControl.lvlib ProductControl.lvlib UI.lvlib PantherLAB_lib_Message.lvlib
Master Reference Not Closed	403687	The "%s" module cannot be run as singleton because the Master Reference is still open from a prior run as cloneable. If you plan on running this module as both singleton and cloneable, consider changing your Main VI to wire a value of TRUE to the 'Close Master Reference' input of Init Module.vi.	PantherLAB_lib_Message.lvlib

Chapter 6. Legal Information

6.1. Document creation

This document has been generated using the following tools.

6.1.1. Antidoc

Project website: Antidoc

Maintainer website: Wovalab

6.1.2. Asciidoc for LabVIEW™

Project website: Asciidoc toolkit

Maintainer website: Wovalab

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.

6.1.3. Graph Builder

Project website: Graph Builder

BSD 3-Clause License

Copyright © 2020, Cyril GAMBINI 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.

6.1.4. classy Diagram Viewer

Project website: classy Diagram Viewer

BSD 3-Clause License

Copyright © 2021, Tatiana Boy 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.

6.2. Product used in the project

The documented project has been developed with the following products.

6.2.1. DQMH®

Copyright © 2021 DQMH® Consortium, LLC. All Rights Reserved.

Find more details on DQMH® Consortium website