Demo Antidoc

Antidoc v3.0.0, LabITStudio

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.

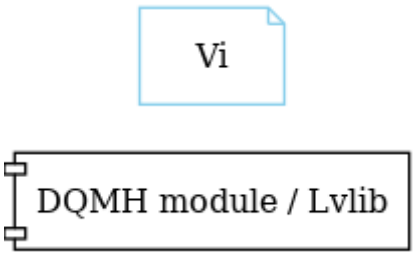
NOTE	Refer to the DQMH® framework official documentation to find more details on how the framework works

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.

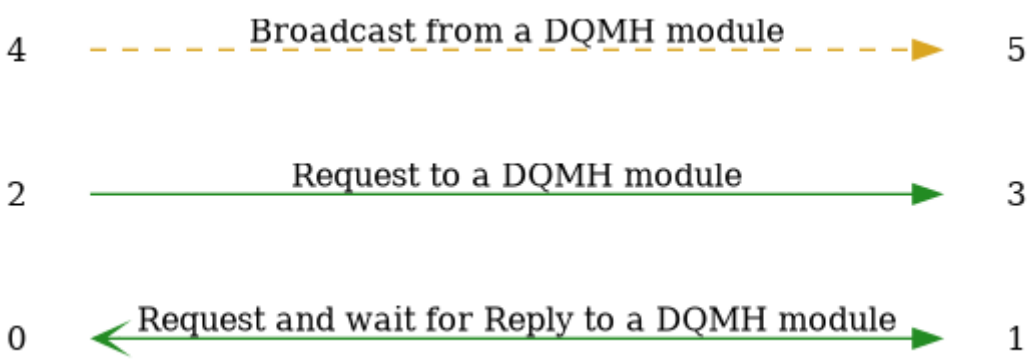
NOTE	If your module has a helper loop, it will be listed along with the events it processes. Helper loops can be categorized as either DQMH or Custom. To ensure your helper loop adheres to DQMH guidelines see the documentaiton.

Graphs used in this section have the following legend:

Components:

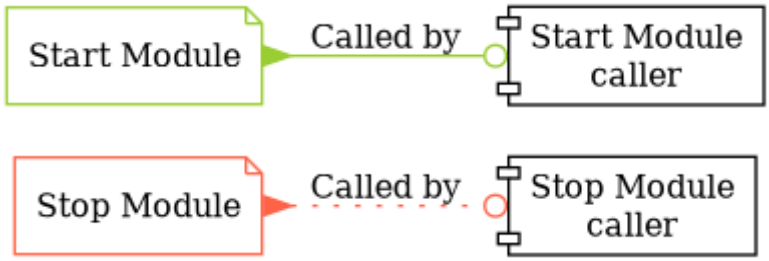


Events:



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 0 singleton module and 1 cloneable module.

Table 1. Modules list

Singleton	Cloneable
	Clon.lvlib

This graph represents the links between all DQMH modules.

2.3. Clon.lvlib

Type: Cloneable

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

2.3.1. Module relationship

Table 2. Requests callers

Request Name	Callers
Do Something Else and Wait for Reply	
Do Something Else	
Do Something	
Hide Panel	
Show Diagram	
Show Panel	

Table 3. Broadcasts Listeners

Broadcast Name	Listeners
Did Something	
Error Reported	
Module Did Init	
Module Did Stop	External Caller.vi
Status Updated	External Caller.vi
Update Module Execution Status	

Table 4. Used requests

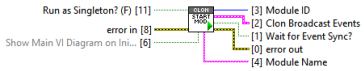
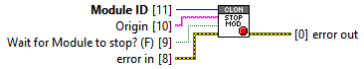
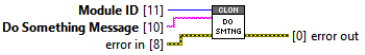
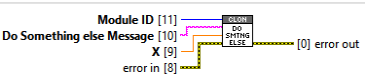
Module	Requests
—	—

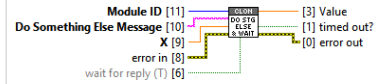
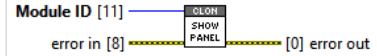
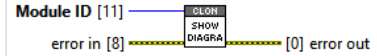
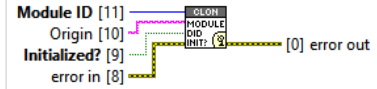
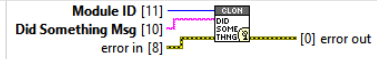
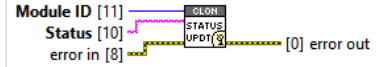
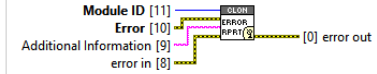
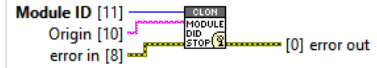
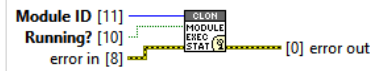
Table 5. Registered broadcast

Module	Broadcasts
—	—

2.3.2. Event list

Table 6. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches 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.			
Do Something			Send the Do something request to the Module's Main.vi.			
Do Something Else			Send the Do Something Else request to the Module's Main.vi.			

Name	Type	Connector pane	Description	S.	R.	I.
Do Something Else and Wait for Reply			Send the Do Something Else request to the Module's Main.vi.			
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.			
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.			
Did Something			Send the Did Something 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

2.3.3. Module Start/Stop calls

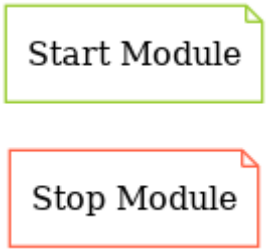


Table 7. Start and Stop module callers

Function	Callers
Start Module	
Stop Module	

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 Clon.lvlib use the following custom errors:

Table 8. Custom errors

Name	Code	Description
Motor Not Running	0	Error information not found in the code
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.
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	

Name	Code	Description
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.

2.3.6. Module Constant VIs

Table 9. Constant VIs Found

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

Chapter 3. Custom errors

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

Table 10. Custom errors

Name	Code	Description	Owned by
Motor Not Running	0		Clon.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.	Clon.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.	Clon.lvlib
Module Not Synced	403683	%s Module was unable to synchronize events.	Clon.lvlib
Module Not Running	403684	Not a single instance of "%s" Module running.	Clon.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.	Clon.lvlib
Request and Wait for Reply Timeout	403686		Clon.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.	Clon.lvlib

Chapter 4. Legal Information

4.1. Document creation

This document has been generated using the following tools.

4.1.1. Antidoc

Project website: Antidoc

Maintainer website: Wovalab

4.1.2. 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.

4.1.3. 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.

4.2. Product used in the project

The documented project has been developed with the following products.

4.2.1. DQMH®

Find more details on DQMH® Consortium website