WATS DQMH Module

Antidoc v2.0.5, Neosoft Technologies Inc.

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.

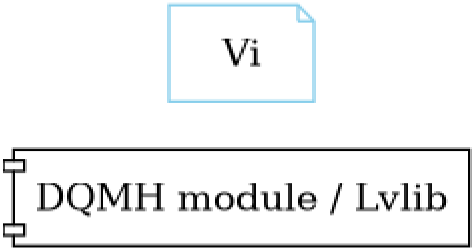
Broadcasts are usually named using past tense or passive voice.

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

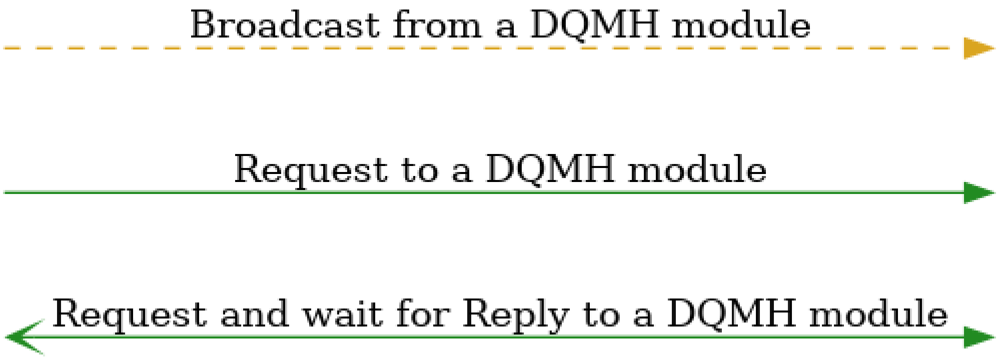
The following section gives you details on the project architecture relying on this framework. It gives you an overview of the modules' interaction and detailed information on each module.

Graphs used in this section have the following legend:

Components:



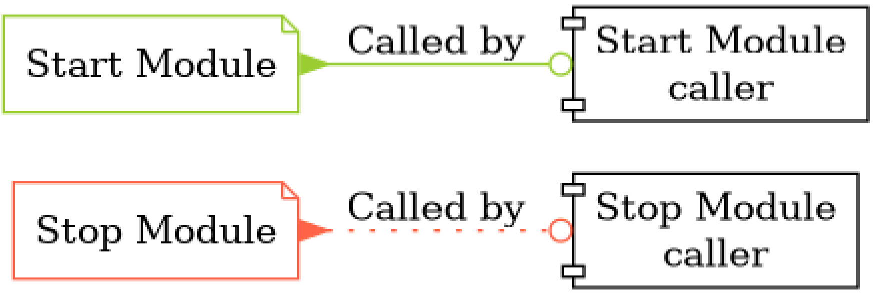
Events:


Broadcast from a DQMH module
Request to a DQMH module
Request and wait for Reply to a DQMH module

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:



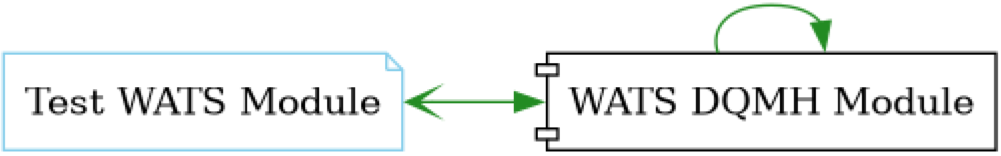
2.2. Modules overview

This project contains 1 singleton module and 0 cloneable module.

Table 1. Modules list

Singleton	Cloneable
WATS DQMH Module.lvlib	

This graph represents the links between all DQMH modules.



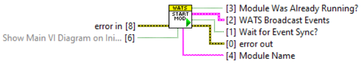
2.3. WATS DQMH Module.lvlib

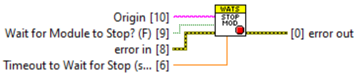
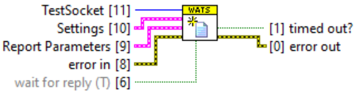
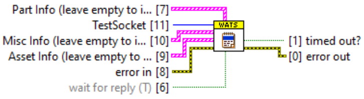
Type: Singleton

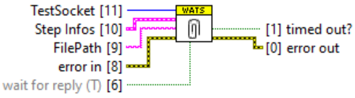
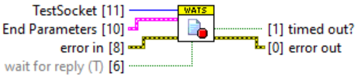
Responsibility: DQMH Module wrapping the WATS toolkit

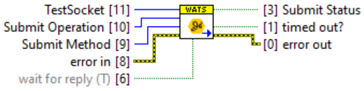
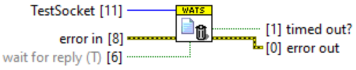
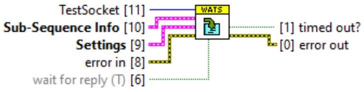
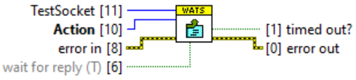
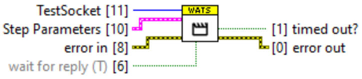
2.3.1. Event list

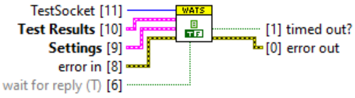
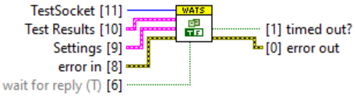
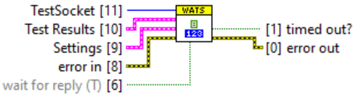
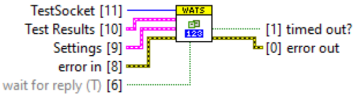
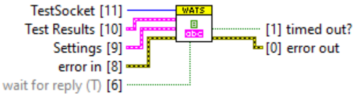
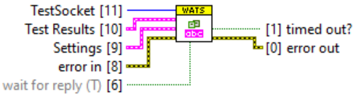
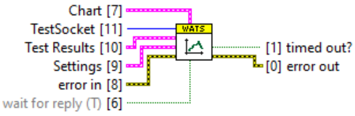
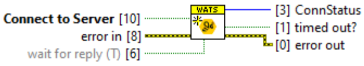
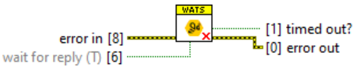
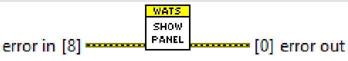
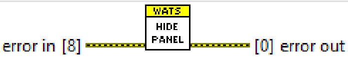
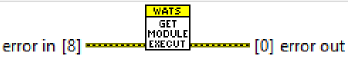
Table 2. Events

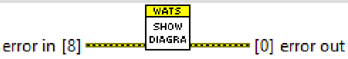
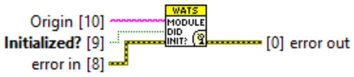
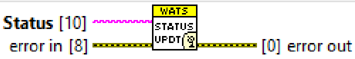
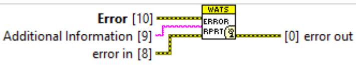
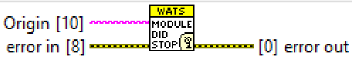
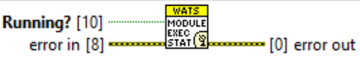
Name	Type	Connector pane	Description	S.	R.	I.
Start Module		 WATS START MOD error in [8] Show Main VI Diagram on Ini... [6] [3] Module Was Already Running? [2] WATS Broadcast Events [1] Wait for Event Sync? [0] error out [4] Module Name	Launches the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a <b>Register For Events</b> function. After the optional Register For Events function call, you should always call the <b>Synchronize Module Events.vi</b> 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 <b>Validate DQMH Module</b> tool added a 'Module Name' output to this VI.			

Name	Type	Connector pane	Description	S.	R.	I.
Stop Module		 Origin [10] Wait for Module to Stop? (F) [9] error in [8] Timeout to Wait for Stop (s... [6] WATS STOP MOD [0] error out	Send the Stop request to the Module's Main.vi. If <b>Wait for Module to Stop?</b> is TRUE, this VI will wait until the module main VI stops, and will timeout at the <b>Timeout to Wait for Stop</b> 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 <b>Timeout to Wait for Stop</b> value is ignored if 'Wait for Module to Stop?' is set to FALSE.			
Start New Report		 TestSocket [11] Settings [10] Report Parameters [9] error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Starts a new report for the specified TestSocket. If there is already an open Report for this TestSocket, use the VI "Submit Report" with 'Submit and Close' or use the VI "Close Report" to delete the report and free up the TestSocket for the new report			
Add Info To Report Header		 Part Info (leave empty to i... [7] TestSocket [11] Misc Info (leave empty to i... [10] Asset Info (leave empty to ... [9] error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Add info to the specified report header : - Part info (sub parts or boards). Used to search relations in WATS reports. - Misc info (like firmware version) - Asset info. Assets must be defined in the Asset Manager in WATS. Asset Serial Number identify the unique asset, Usage Count will increase the running count on the asset. * Leaving an array empty will not add this type of info to the report.			

Name	Type	Connector pane	Description	S.	R.	I.
Attach File to Report		 TestSocket [11] Step Infos [10] FilePath [9] error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Attach a file to the specified report. <b>ctrl.Action</b> Standard step input <b>File Path:</b> Enter full path including file name (c:\folder1\filename.ext) <b>Delete File:</b> If true, the VI will delete the file after attached to the WATS report Depending on the client version running, the file size is limited to a default size.			
End Report		 TestSocket [11] End Parameters [10] error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Optionally used to mark the end of the specified report. Any inputs to will overwrite the parameters entered by Start New Report request.			

Name	Type	Connector pane	Description	S.	R.	I.
Submit Report		 TestSocket [11] Submit Operation [10] Submit Method [9] error in [8] wait for reply (T) [6] WATS [3] Submit Status [1] timed out? [0] error out	<b>Select Type input</b> <b>Submit UUT:</b> Submit UUT report to server and keep LabVIEW references open for later use when submiting UUR report <b>Submit UUT and Close:</b> Submit UUT report to server and close off LabVIEW references. Frees up TestSocket for new report <b>Submit UUR:</b> Submit UUR report and close off LabVIEW references. Frees up TestSocket for new report <b>Submit Method input</b> <b>Automatic:</b> Let the API determine state of connection to server and submit methods (Synchronous/Asynchronous) <b>Synchronous (Online):</b> Submits the UUT report direct to the server. The VI will not exit until the transfer of the report is completed and confirmed by the server. This method is more time consuming that Asynchronous. <b>Asynchronous (Offline):</b> RECOMMENDED. The VI will call a "Submit" method to the WATS API and the WATS Client will make sure the Report is securly transferred to the server. Does support offline logging.			
Close Report		 TestSocket [11] error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Deletes the report linked to the specified TestSocket without sending it to the WATS server. Frees up the TestSocket for another report.			
Add Sub-Sequence		 TestSocket [11] Sub-Sequence Info [10] Settings [9] error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Add a sub-sequence to the specified report.			
Change Active Sequence Level		 TestSocket [11] Action [10] error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Change the level of the active sequence of the specified report. Either go up one sequence level, or go back to the top level sequence.			
Add Action Step		 TestSocket [11] Step Parameters [10] error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Add an action step to the specified report			

Name	Type	Connector pane	Description	S.	R.	I.
Add Pass-Fail Test		 TestSocket [11] Test Results [10] Settings [9] error in [8] wait for reply (T) [6] WATS TF [1] timed out? [0] error out	Add the results of a Pass/Fail test to the specified report			
Add Pass-Fail Test - Multiple		 TestSocket [11] Test Results [10] Settings [9] error in [8] wait for reply (T) [6] WATS TF [1] timed out? [0] error out	Add the results of a Pass/Fail test, containing multiple pass/fail results, to the specified report			
Add Numeric Limit Test		 TestSocket [11] Test Results [10] Settings [9] error in [8] wait for reply (T) [6] WATS 123 [1] timed out? [0] error out	Add a numeric limit test to the specified report			
Add Numeric Limit Test - Multiple		 TestSocket [11] Test Results [10] Settings [9] error in [8] wait for reply (T) [6] WATS 123 [1] timed out? [0] error out	Add the results of a Numeric Limit test, containing multiple results, to the specified report			
Add String Value Test		 TestSocket [11] Test Results [10] Settings [9] error in [8] wait for reply (T) [6] WATS abc [1] timed out? [0] error out	add string value test to the specified report			
Add String Value Test - Multiple		 TestSocket [11] Test Results [10] Settings [9] error in [8] wait for reply (T) [6] WATS abc [1] timed out? [0] error out	Add the results of a String Value test, containing multiple results, to the specified report			
Add XY Graph Multiple Numeric Limit Test		 Chart [7] TestSocket [11] Test Results [10] Settings [9] error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Add a XY Graph test to the specified Report. Example of a XY Graph test is a FFT measurement. The FFT curve will be shown as a graph in WATS. A test on the FFT could be 1st harmonic and 2nd harmonic, where 1st harmonic should be within 2-3 kHz and 2nd harmonic should be within 4-6 kHz.			
Initialize WATS		 Connect to Server [10] error in [8] wait for reply (T) [6] WATS [3] ConnStatus [1] timed out? [0] error out	Create TDM Object and optionally connect to the server. If not connected, all reports will be temporary stored to a file share and transferred when online.			
Close WATS		 error in [8] wait for reply (T) [6] WATS [1] timed out? [0] error out	Close and destroy WATS object.			
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.			

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).			
Module Did Init	📡	 WATS MODULE DID INIT? Origin [10] Initialized? [9] error in [8] [0] error out	Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Status Updated	📡	 WATS STATUS UPDT Status [10] error in [8] [0] error out	Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported	📡	 WATS ERROR RPRT Error [10] Additional Information [9] error in [8] [0] error out	Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop	📡	 WATS MODULE DID STOP Origin [10] error in [8] [0] error out	Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status	📡	 WATS MODULE EXEC STAT Running? [10] error in [8] [0] error out	Broadcast event to specify whether or not the module is running.			

Type: ⚬➝ → Request | ⚬➝↺ → Request and Wait for Reply | 📡 → Broadcast

Scope: 🔑 → Protected | 🔑 → Community

Reentrancy: Ⓟ → Preallocated reentrancy | Ⓢ → Shared reentrancy

Inlining: ⇉ → Inlined

2.3.2. Module relationship

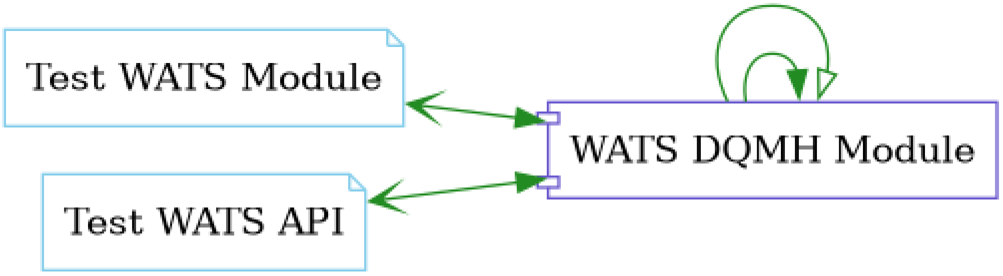


Table 3. Requests callers

Request Name	Callers
Add Action Step	Test WATS API.vi Test WATS Module.vi
Add Info To Report Header	Test WATS API.vi Test WATS Module.vi
Add Numeric Limit Test - Multiple	Test WATS API.vi Test WATS Module.vi
Add Numeric Limit Test	Test WATS API.vi Test WATS Module.vi
Add Pass-Fail Test - Multiple	Test WATS API.vi Test WATS Module.vi
Add Pass-Fail Test	Test WATS API.vi Test WATS Module.vi
Add String Value Test - Multiple	Test WATS API.vi Test WATS Module.vi
Add String Value Test	Test WATS API.vi Test WATS Module.vi
Add Sub-Sequence	Test WATS API.vi Test WATS Module.vi
Add XY Graph Multiple Numeric Limit Test	Test WATS API.vi
Attach File to Report	Test WATS API.vi Test WATS Module.vi
Change Active Sequence Level	Test WATS API.vi Test WATS Module.vi
Close Report	Test WATS API.vi
Close WATS	Test WATS API.vi Test WATS Module.vi
End Report	Test WATS API.vi
Get Module Execution Status	WATS DQMH Module.lvlib:Obtain Broadcast Events for Registration.vi WATS DQMH Module.lvlib:Start Module.vi
Hide Panel	Test WATS API.vi
Initialize WATS	Test WATS API.vi Test WATS Module.vi
Show Diagram	Test WATS API.vi
Show Panel	Test WATS API.vi
Start New Report	Test WATS API.vi Test WATS Module.vi

Request Name	Callers
Submit Report	Test WATS API.vi Test WATS Module.vi

Table 4. Broadcasts Listeners

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

Table 5. Used requests

Module	Requests
WATS DQMH Module.lvlib	Stop Module.vi

Table 6. Registered broadcast

Module	Broadcasts
—	—

2.3.3. Module Start/Stop calls

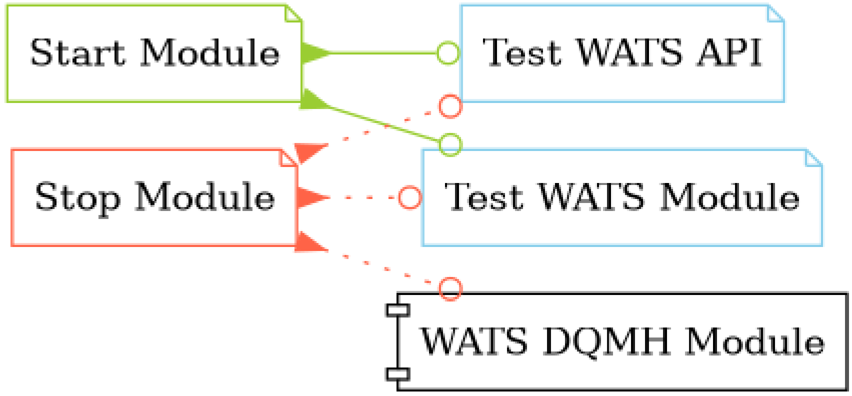


Table 7. Start and Stop module callers

Function	Callers
Start Module	Test WATS Module.vi Test WATS API.vi
Stop Module	Test WATS Module.vi WATS DQMH Module.lvlib:Handle Exit.vi Test WATS API.vi

2.3.4. Module custom errors

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

Module WATS DQMH Module.lvlib use the following custom errors:

Table 8. Custom errors

Name	Code	Description
Module Not Running	0	
Module Not Stopped	0	
Module Not Synced	0	
Request and Wait for Reply Timeout	0	

Chapter 3. Custom errors

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

Table 9. Custom errors

Name	Code	Description	Owned by
Module Not Running	0		WATS DQMH Module.lvlib
Module Not Stopped	0		WATS DQMH Module.lvlib
Module Not Synced	0		WATS DQMH Module.lvlib
Request and Wait for Reply Timeout	0		WATS DQMH Module.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

4.1.3. Graph Builder

Project website: Graph Builder

BSD 3-Clause License

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