Glasswall Dynamic Configuration

© 2014 Glasswall Solutions Ltd ALL RIGHTS RESERVED

Information contained herein is the property of Glasswall Limited and is proprietary and confidential.

Paul Gerard

Glasswall Solutions Ltd.

(e): support@glasswallsolutions.com

Creation Date – 08/05/2014

Version – A

**Document History**

Table 1: Document Change History

|  |  |  |  |
| --- | --- | --- | --- |
| *Issue Date* | *Issue Number* | *Author* | *Description* |
| 08/05/2014 | A | PFG | Initial version for review |
|  |  |  | . |

**Document Distribution**

Table 2: Document Distribution

|  |  |  |  |
| --- | --- | --- | --- |
| *Name* | *Position* | *Date* | *Signature* |
| Glasswall Development Team |  |  | . |
|  |  |  | . |

**Copyright and Contact Details**

The copyright in this work is vested in Glasswall Solutions Ltd, and the document is issued in confidence for the purpose for which it is supplied. It must not be reproduced in whole or in part or used for tendering or manufacturing purposes except under agreement or with the consent in writing of Glasswall Solutions Limited and then only on condition that this notice is included in any such reproduction. No information as to the contents or subject matter of this document or any part thereof arising directly or indirectly there from shall be given orally or in writing or communicated in any manner whatsoever to any third part being an individual firm or company or any employee thereof without the prior consent in writing of Glasswall Solutions Limited.

© Glasswall Solutions Limited 2014

If there are any questions related to this report, these should be addressed to:

Sam Hutton

Glasswall Solutions Limited

e-mail: [shutton@glasswallsolutions.com](mailto:shutton@glasswallsolutions.com)

**Index**

[1 Introduction 4](#_Toc387311106)

[1.1 Glossary 4](#_Toc387311107)

[1.2 References 4](#_Toc387311108)

[1.3 Schematics 4](#_Toc387311109)

[2 Use Cases 6](#_Toc387311110)

[2.1 Start-up 6](#_Toc387311111)

[2.2 Setting Configuration Items 6](#_Toc387311112)

[2.3 Reading Current Configuration 6](#_Toc387311113)

[2.4 Clearing Configuration Updates 6](#_Toc387311114)

[3 Components 7](#_Toc387311115)

[3.1 Overview 7](#_Toc387311116)

[3.2 Glasswall Top Level 8](#_Toc387311117)

[3.3 Utils 8](#_Toc387311118)

[3.3.1 Configuration Repository Module 8](#_Toc387311119)

[3.3.2 Configuration Loader Module 9](#_Toc387311120)

[3.3.3 Configuration Exporter Module 9](#_Toc387311121)

[3.3.4 GW XML Config Module 10](#_Toc387311122)

[3.4 Camera Configuration 10](#_Toc387311123)

[4 Key Scenarios 12](#_Toc387311124)

[4.1 Start-up 12](#_Toc387311125)

[4.2 Configuration Item Update 12](#_Toc387311126)

[4.3 Configuration Export 13](#_Toc387311127)

# Introduction

This document provides an overview of the functionality associated with the Glasswall Dynamic Configuration API.

The Use Cases section describes the functionality that the feature provides.

The Components section describes the architecturally significant modules associated with the feature and their interfaces.

The Key Scenarios section provides sequence diagrams of key use cases, demonstrating the interaction between the principle components.

## Glossary

The following terms are used through this document.

The Library – the Glasswall library

The Client Application – application using the library

The API – the application programming interface (API) used by the client when interfacing with the library.

The Configuration – settings used by the library that may be updated by the client application.

A configuration item – The Configuration is made up of sets of tuneable values that are associated with the various document cameras. Each configuration item is identified be specifying the camera and the configuration item’s name.

## References

*Content Management Configuration XSD*

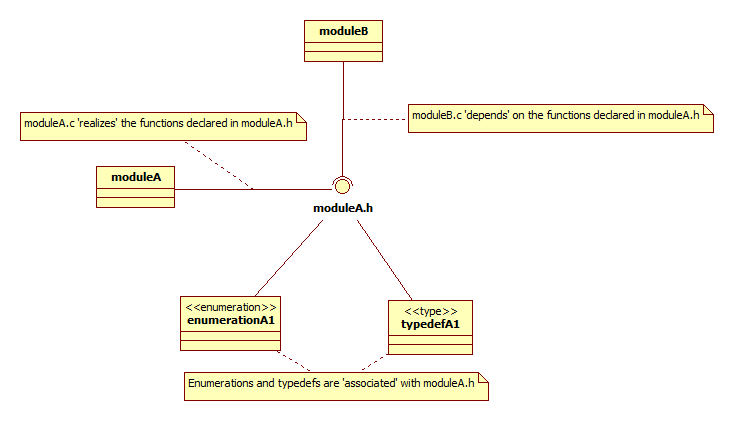
$Glasswall\glasswall.classic\documentation\doxygen\xsdFiles\ gwContentManagementConfiguration.xsd

This XSD defines the format of the XML used to configure the Glasswall DLL. This is the same definition used to format the configuration when exported from the DLL.

## Schematics

The WhiteStarUML design tool has been used to document the design of this feature. The diagrams in this document follow the guidelines shown in the diagram below. The diagram represents the following relationships.

* The functions declared in moduleA.h are defined in moduleA.c.
* The file moduleB.c ‘#include’s moduleA.h.
* The enumeration and typedef (represented using the stereotypes) are defined in moduleA.h



# Use Cases

The following sections capture the high-level functionality of this feature.

## Start-up

From the Client Application having loaded the Library, it will be configured with a set of default configuration.

*Default settings for content management flags is ‘sanitise’.*

*Default setting for watermark is ‘Glasswall Approved’.*

## Setting Configuration Items

From the Client Application all configuration items in all cameras can be updated in one call to the API

From the Client Application an individual configuration item can be updated through the API.

From the Client Application updating a configuration item in one camera, does not affect the value of a configuration item of the same name in a different camera.

From the Client Application, attempting to configure a content item that does not exist will result is an error status and message being reported.

*Explanatory error messages are retrievable through the API.*

From the Client Application attempting to update a configuration item with a non-valid value will result in an error status and message being reported.

From the Client Application any changes in the configuration will be used in all subsequent calls through the API.

## Reading Current Configuration

From the Client Application, all configuration can be retrieved from the Library with one call to the API.

*The Configuration will be returned in an xml string that is in the same format as used in the configuration operation.*

From the Client Application, retrieving the configuration from the Library will not require any memory management.

## Clearing Configuration Updates

From the Client Application, the Configuration can be returned to the default values set at start-up with one call to the API.

# Components

This section provides an architectural description of the solution.

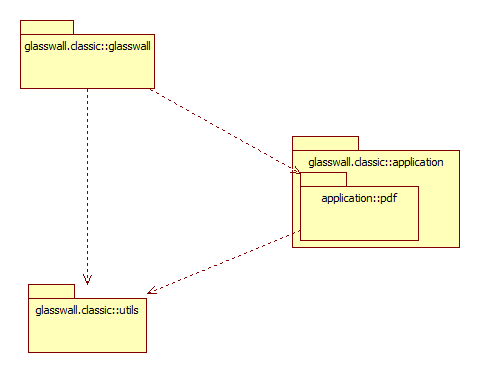
## Overview

The principle components involved in the configuration of the Glasswall library are shown in the diagram below. The scoping operator :: is used to indicate the nesting of components in order to provide some level of context only, no other semantic interpretation should be applied.

The component, glasswall.classic::glasswall, consists of the top layer of the library with modules that implement the client accessible interface of the library.

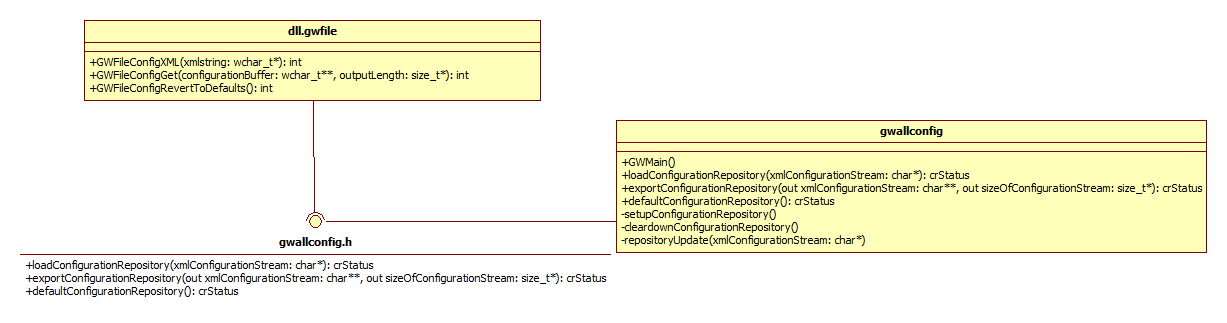
The component, glasswall.classic::utils, provides utility functionality to the interface layer and the cameras.

The component, application::pdf, provides the PDF camera functionality. This camera is responsible for the processing of PDF documents.



## Glasswall Top Level

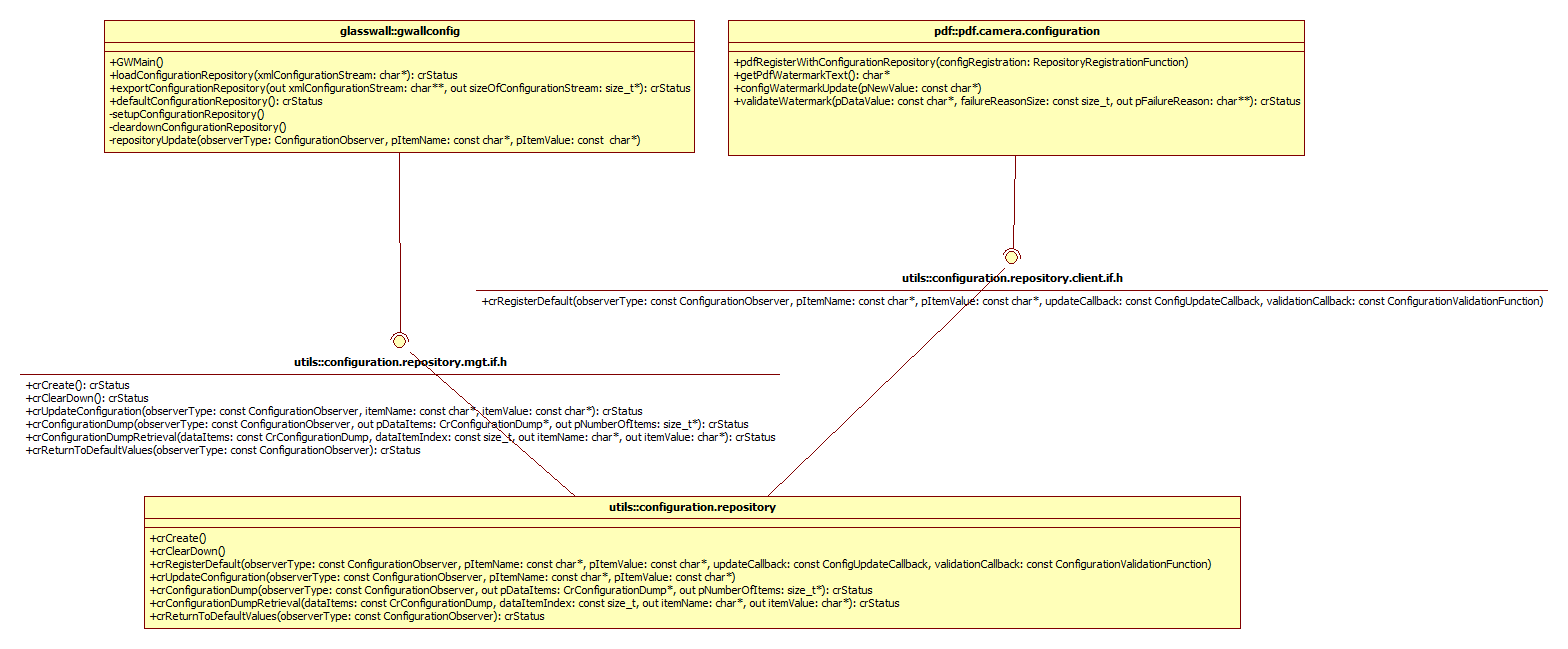
The library’s configuration is managed through the API, in glasswall.classic::glasswall. The dll.dwfile module provides the API functionality, delegating responsibility of interfacing with the configuration repository to the gwallconfig module.



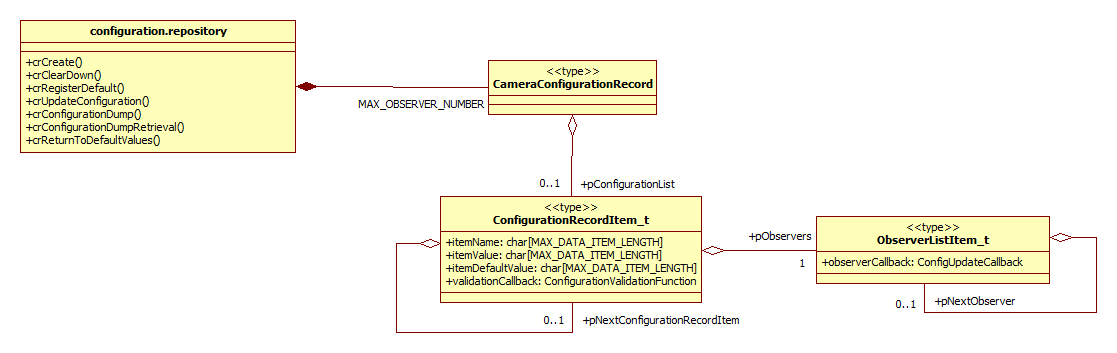
## Utils

### Configuration Repository Module

The configuration repository is the data storage container for Glasswall dynamic configuration. The module forms part of the glasswall.classic::utils component. The module provides two interfaces, one supporting the management aspects of the module, the other providing functionality to client modules. This split is shown in the diagram below.

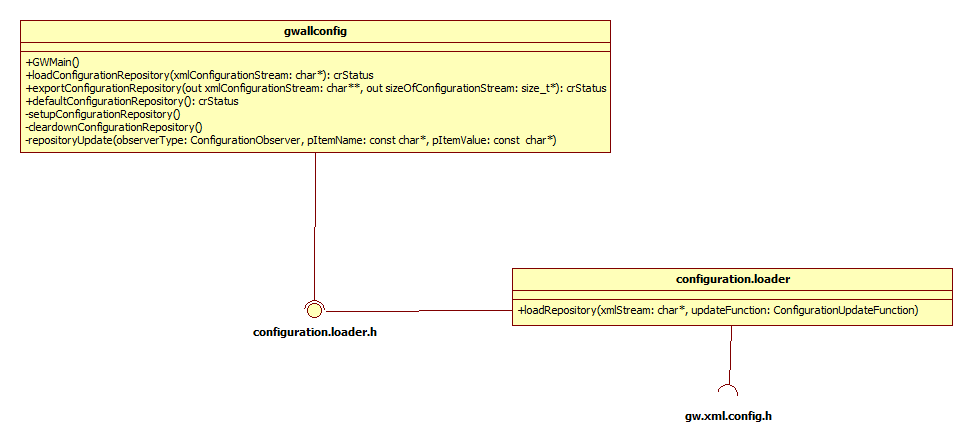


The data structure used by the configuration.repository module to store the configuration and associated observers is shown in the diagram below.



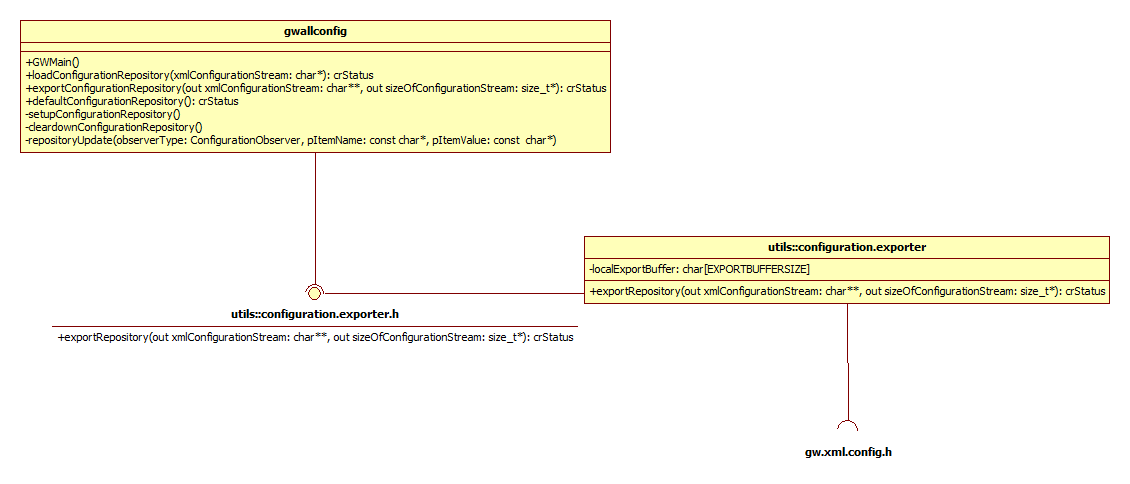
### Configuration Loader Module

The configuration repository is updated from the client application using the configuration.loader module. This module is responsible for decoding the supplied XML string and updating the repository.



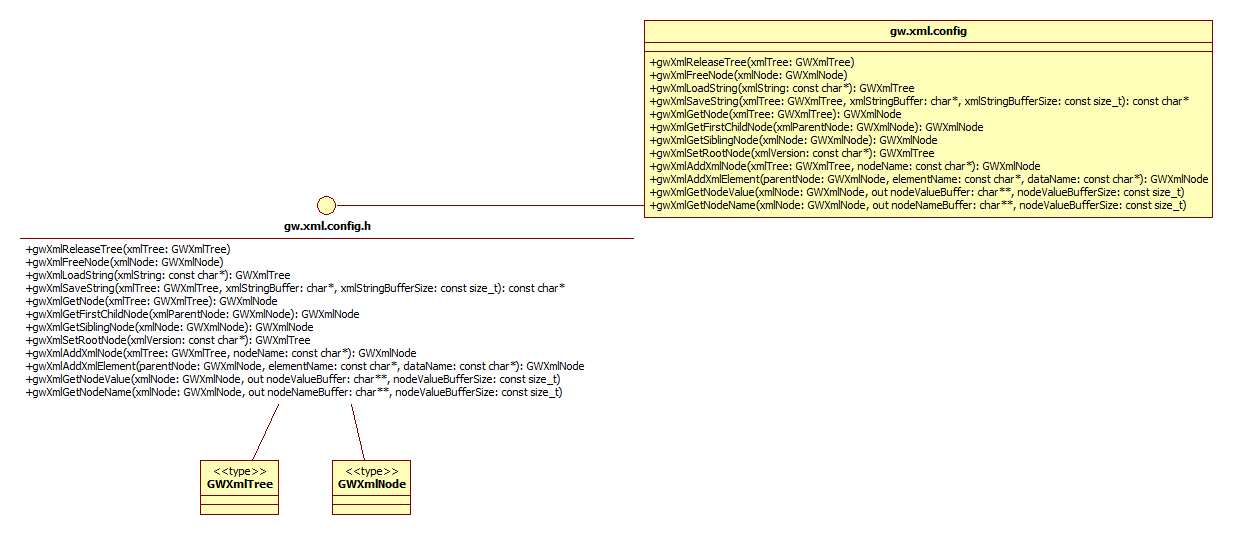
### Configuration Exporter Module

To support the export of configuration to the client application the gwallconfig module makes use of the configuration.export module. The configuration.export module has responsibility for formatting the contents of the configuration repository into the required XML format.



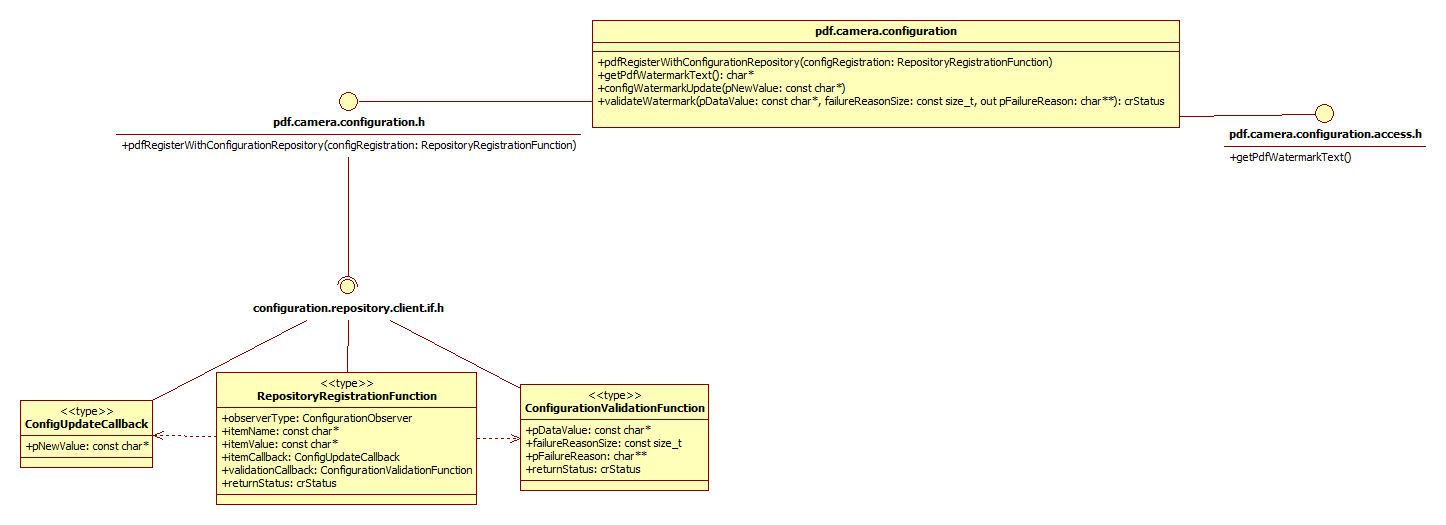
### GW XML Config Module

The Configuration Loader and Configuration Exporter modules have responsibility associated with the receiving and publishing of Glasswall configuration in XML format. These modules use the GW XML Config module to access XML functionality. The GW XML Config module is a wrapper around the MXML third-party library.



## Camera Configuration

As shown in section 3.3 above the cameras have a dependency on the configuration.repository.client.if.h interface. Through this interface they register their configuration with the configuration repository. The camera’s configuration module, in this example pdf.camera.configuration, is responsible for providing a function used by gwallconfig to instigate the registration of the camera’s configuration with the repository. In the diagram below only the functions associated with the PDF Watermark configuration have been included as examples.



As part of the registration process the camera’s configuration module registers the data item name and the default data item value. Along with these configuration items the configuration module also provides an update callback function and a validation callback function.

The data item’s validation callback function is called when an attempt is made to updated the value of the data item to ensure that the new value is valid. The

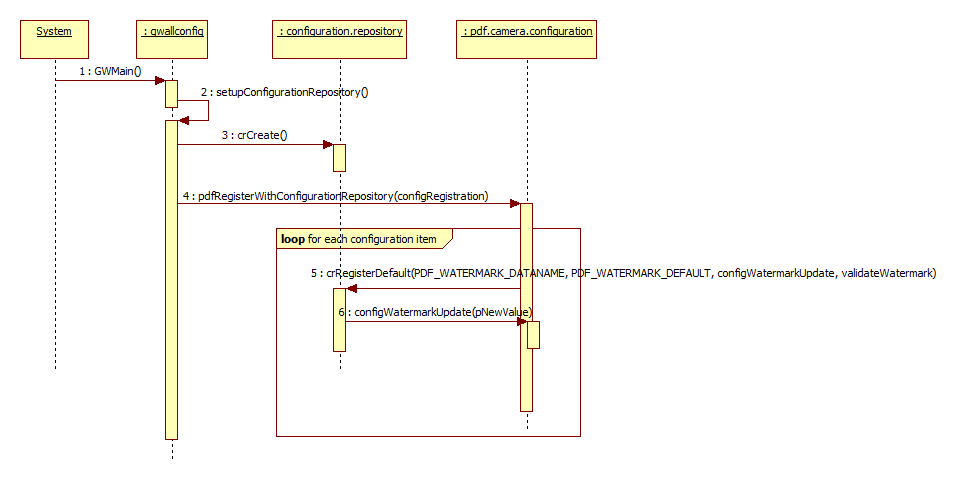
The data item’s update callback function is called by the configuration repository in the event of the data item’s value being changed. The callback is then responsible for ensuring that the updated value is made available to the camera.

Other modules within the PDF camera access the PDF Camera Configuration through the pdf.camera.configuration.access.h interface.

# Key Scenarios

## Start-up

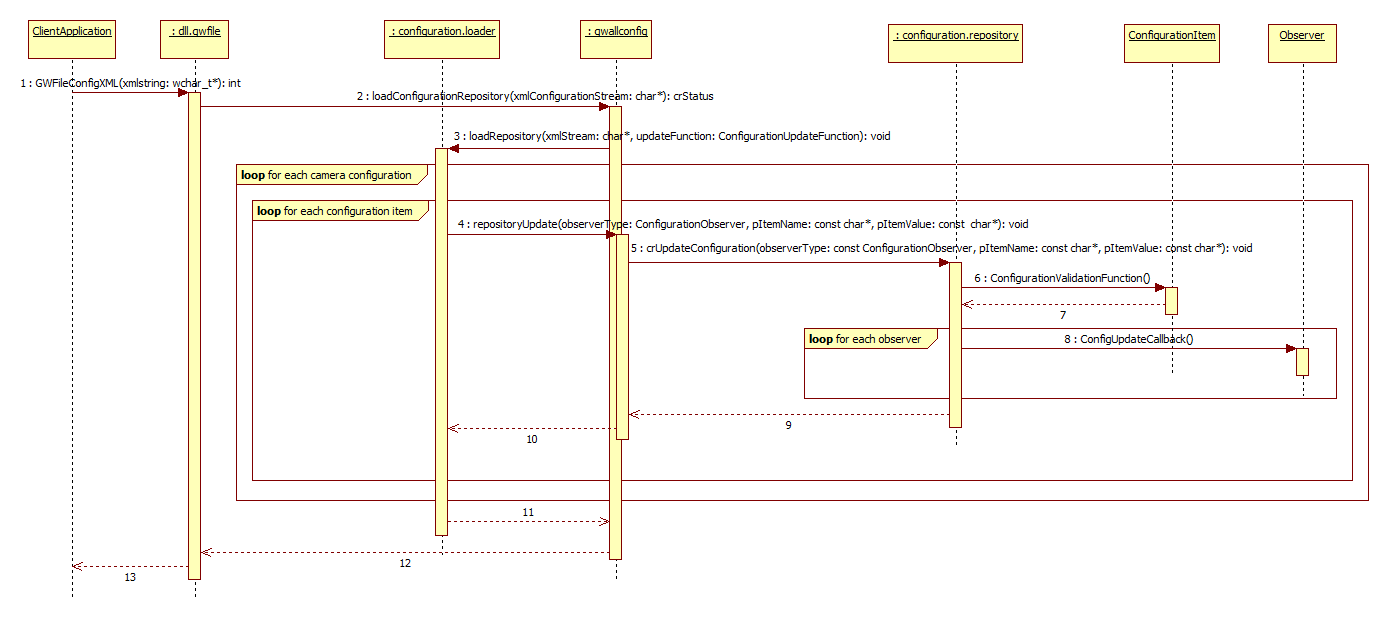
The call from the System object in the sequence diagram below is implemented using platform specific techniques, encapsulated within GWMain. All sequencing from GWMain is common between platforms. In common with earlier examples, only the PDF Watermark configuration item is included for simplicity. In the event of addition configuration items being used by a camera then multiple calls to crRegisterDefault would be made by the camera.



The addition of further cameras to the configuration system would involve the gwallconfig module’s setupConfigurationRepository function being extended to include the additional cameras’ ‘RegisterWithConfigurationRepository’ functions.

## Configuration Item Update

The GWFileConfigXML API function provide the client application with the ability to modify the Glasswall configuration. It is not necessary to submit a complete XML configuration string to the function, only those configuration items which require updating need be included.

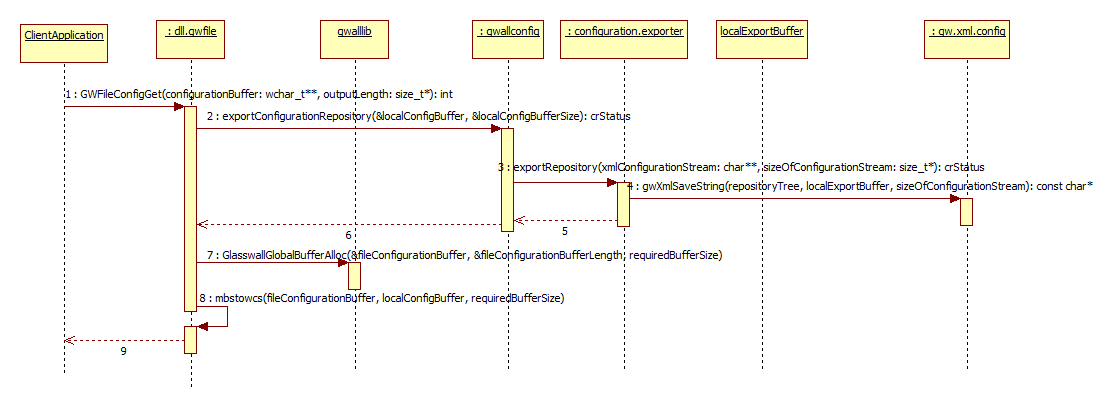


When the XML configuration string (see definition in References section) is received by the gwallconfig module it uses the configuration.loader to parse the string and update the repository using the specified update function.

When a configuration item is passed to the configuration repository for update the new value if validated using the function provided at start-up, if successful any observers of that configuration item are informed using the callback defined when the observer was registered.

## Configuration Export

The export of configuration from the repository requires storage to be made available to the client application. To avoid the client application having responsibility for correctly releasing resources the existing global buffer facility (provided by the gwalllib module) is used to provide managed resources that can safely be provided to the client application.



Since the Global Buffer Allocation requires a size to be allocated, a fixed size buffer (localExportBuffer in the above sequence) is used by the configuration.exporter module to store the initial XML string. The location of localExportBuffer is then returned to dll.gwfile and used as the source string when sizing and populating the globally allocated buffer.