

Glasswall API Language Wrapper Documentation

© 2020 Glasswall Solutions Ltd ALL RIGHTS RESERVED

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

Glasswall Solutions Ltd.

[support@glasswallsolutions.com](mailto:support%40glasswallsolutions.com)

Creation Date – 14 October 2020

Version – 1

**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 2020

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

Glasswall Solutions Limited

e-mail: [support@glasswallsolutions.com](mailto:support%40glasswallsolutions.com)

**Index**

[1. Introduction 4](#_Toc53561975)

[1.1 Issues to consider during implementation 4](#_Toc53561976)

[2. JavaScript 5](#_Toc53561977)

[2.1 Files provided 5](#_Toc53561978)

[2.2 Dependencies 5](#_Toc53561979)

[2.2.1 Framework dependencies 5](#_Toc53561980)

[2.2.2 Module dependencies 5](#_Toc53561981)

[2.3 Wrapper integration 5](#_Toc53561982)

[2.4 Code example 5](#_Toc53561983)

[2.5 Issues to consider 6](#_Toc53561984)

# Introduction

This is an introductory guide on how to use the language wrappers for the Glasswall API.

For each language we have provided the following:

* The required dependencies and the target platform or framework required for the language. The mentioned platforms and frameworks are the ones that were used during testing, but other versions may or may not work.
* A general overview on integrating Glasswall into a project.
* A code example demonstrating how the Glasswall library can be used to process a directory of files. Each example shows the file being processed in Manage and Protect mode as well as being analysed in Analysis mode.

## Issues to consider during implementation

* The Glasswall library is not thread safe, which means that the language wrappers are not thread safe. This can be overcome by running the Glasswall library in a separate process.
* We recommend that the Glasswall library is run in separate process in case unforeseen issues arise.

# JavaScript

## Files provided

The JavaScript wrapper is provided as a single JavaScript file that you include in your script.

glasswall.classic.javascript.js – The JavaScript file containing the Glasswall module that is used to interact with the Glasswall library.

## Dependencies

### Framework dependencies

The JavaScript wrapper requires Node.js 8 to be installed.

### Module dependencies

The JavaScript wrapper depends on the following npm modules:

* ffi – 2.20
* ref – 1.3.5
* ref-wchar – 1.02

A JSON file with all specific dependencies required is deployed with the SDK wrapper, therefore the npm install command can be executed to automatically acquire all required dependencies.

## Wrapper integration

The wrapper can be integrated into an existing project by adding the module dependencies and then calling the **require** function from your JavaScript script. The path to the Glasswall library is then passed in as an argument when constructing a Glasswall object.

It is possible to install the modules in an offline environment with the Yarn package manager. More information can be found at <https://yarnpkg.com/blog/2016/11/24/offline-mirror/>

## Code example

|  |
| --- |
| let fs = require('fs');  let path = require('path')  let getFiles = function (dir, filelist) {  let files = fs.readdirSync(dir);  filelist = filelist || [];  files.forEach(function (file) {  if (fs.statSync(dir + "/" + file).isDirectory()) {  filelist = getFiles(dir + '/' + file, filelist);  }  else {  filelist.push(dir + "/" + file);  }  });  return filelist;  };  let writeToFile = function (outDirectory, fileName, content) {  fs.writeFileSync(outDirectory + "/" + fileName, content, function (err) {  if (err)  return console.log(err);  });  };  let main = function () {  const args = process.argv;  let inputDirectory = args[2]; // The input directory  let outputDirectory = args[3]; // The output directory  let pathToConfig = args[4]; // The path to the XML content management configuration  let pathToGwLib = args[5]; // The path to the Glasswall library  let glasswall = require("./glasswall.classic.javascript.js");  console.log("Loading Library...");  let gw = new glasswall(pathToGwLib);  let xmlContent = fs.readFileSync(pathToConfig);  // Apply the XML content management configuration  let configXMLStatus = gw.GWFileConfigXML(xmlContent.toString());  if (configXMLStatus != 1) {  // Print the failure reason  console.log("Failed to apply the content management configuration for the following reason: " + gw.GWFileErrorMsg());  return;  }  fs.mkdirSync(outputDirectory);  let files = getFiles(inputDirectory);  for (let i in files) {  let filePath = files[i];  let extension = path.extname(filePath).substring(1);  let filename = path.basename(filePath);  // Process the file in File to Memory Protect  let manageAndProtectResult = gw.GWFileProtect(filePath, extension);  if (manageAndProtectResult.status === 1) {  writeToFile(outputDirectory, filename, manageAndProtectResult.fileBuffer);  }  // Analyse the file in File to Memory Analysis  let analysisResult = gw.GWFileAnalysisAudit(filePath, extension);  console.log(analysisResult.status);  writeToFile(outputDirectory, filename + ".xml", analysisResult.xmlReport);  }  gw.GWFileDone();  };  if (require.main === module) {  main();  } |

## Issues to consider

* The JavaScript wrapper has not been tested on Node.js 10.