# Glasswall Core 2 Wrapper Documentation

# JavaScript

## Purpose

The purpose of the wrappers is to expose the Core 2 SDK functionality through Python, C#, JavaScript and Java.

Each wrapper consists of:

* The wrapper itself: a bridge between the Core 2 SDK and the desired language.
* A series of supporting files (language dependent).

## General Requirements

The following general requirements must be met to use the wrappers and their test apps:

* The Glasswall Core 2 libraries and their dependencies
  + glasswall\_core2.dll
  + \*\_camera.dll
* A designated folder containing files to be input into Core 2
* A designated folder to hold the output from Core 2
* A policy file to modify the default Core 2 file processing behaviour
* The wrapper itself.

## Test Application Overview

Each wrapper is provided with a test application. This application is designed to call each of the Core2 APIs from the chosen language and generate a log file of the results.   
The execution steps are:

* All supporting files, folders and dependencies are checked
* The contents of the output directory are erased in preparation for file processing
* For each file in the input directory:
  + A new folder is created in the output directory and is named for the file currently being processed
  + A series of 23 tests are performed, as detailed in the Wrapper Test Calls document
  + The files generated by Glasswall are saved in the specified output directory
  + A log file detailing the result of each test is generated and saved as local\_process\_log.txt
* When all files have been processed, a final log file named process\_log.txt is saved in the root of the output directory.

## JavaScript Wrapper

### Environment

* The JavaScript wrapper requires Node.js version 10.16.3. More recent versions of Node.js may not be compatible.
* Additional required modules are
  + node-ffi (Node.js Foreign Function Interface). The Node.js addon for loading and calling dynamic link libraries.
  + ref, the native addon for Node.js. This extends the Buffer class.
  + The node\_modules directory contents, as supplied.

Ensure the Core 2 libraries and all required folders are accessible to the wrapper.

### Example Code

The following code uses the JavaScript wrapper to process a file and place the managed file in a buffer. A policy file is specified, and an analysis report is generated. Note that the memory buffer is non-persistent and will have to be processed, analysed, or stored, before the script finishes. The policies file, config\_sanitise.xml will be placed in the specified directory.

|  |
| --- |
| const fs = require('fs');  const path = require('path');  const ref = require('ref');    function main() {    let glasswall = require("c:\\demo\\Core2JS.js");  let gw = new glasswall("e:\\demo\\glasswall\_core2.dll");    try {  // open session  let session\_id = gw.GW2OpenSession();    // register inputfile  let return\_status = gw.GW2RegisterInputFile(session\_id, 'e:\\TestFiles\\C.bmp');    // register outputmemory: the processed data is non-persistent  let output\_file\_buffer = ref.alloc(ref.refType(ref.types.CString));  let output\_buffer\_size = ref.alloc(ref.types.size\_t, 0);  return\_status = gw.GW2RegisterOutputMemory(session\_id, output\_file\_buffer,  output\_buffer\_size);    // register policies file  return\_status = gw.GW2RegisterPoliciesFile(session\_id,  'e:\\Tasks\\W57240-Core\_2\_JavaScript\_Wrapper\\Src\\config\_sanitise.xml', 0);    // register analysis file  return\_status = gw.GW2RegisterAnalysisFile(session\_id,  'e:\\Output\\Analysis\_output.xml', 0); // 0 = AF\_XML    // run the session  return\_status = gw.GW2RunSession(session\_id);    // close the session  return\_status = gw.GW2CloseSession(session\_id);    }  catch (err) {  console.log(`error - exception caught: ${err}`);  process.exit();  }  }    if (require.main === module) {  main();  } |