# Glasswall Core 2 Security Tagging Wrapper Documentation

## Purpose

The purpose of the wrappers is to expose the Core 2 Security Wrapper 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

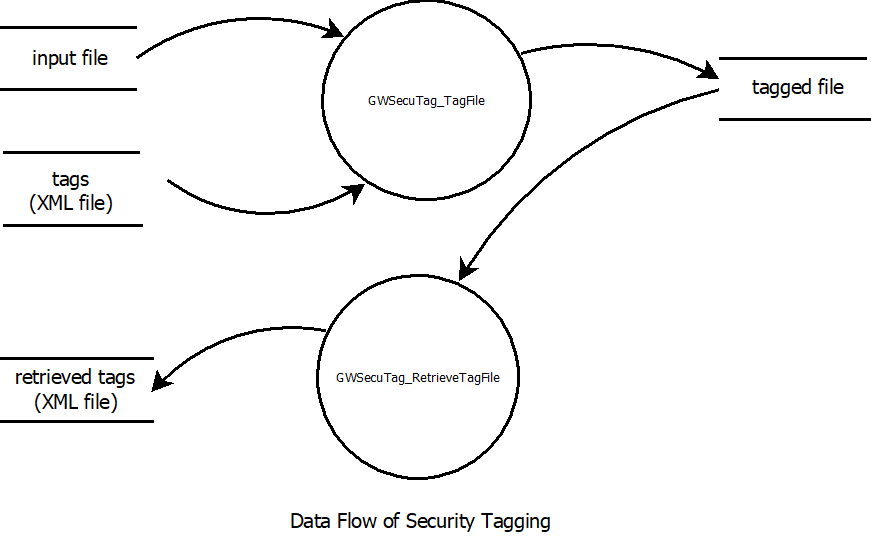
The following general requirements must be met to use the wrappers and their test apps:

* gw\_securtag.dll
* \*\_camera.dll
* Qt5Core.dll
* Qt5Network.dll
* Qt5Xml.dll
* Qt5XmlPatterns.dll
* Quazip.dll

## Test Application Overview

Each wrapper is provided with a test application. This application is designed to call each of the Core 2 Security Tagging APIs from the chosen language.

The process is described in this Data Flow Diagram:



## JavaScript Wrapper

### Environment

* The JavaScript wrapper requires Node.js version 10.16.3. More recent versions of Node 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 General Requirements are met and that 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 tagged file in the specified file. The code then reads from that file and extracts the tags and places them in the specfiied textracted tags file. config\_sanitise.xml will be placed in the specified directory.

|  |
| --- |
| // Demonstrate calls to the Security Taging wrapper using JavaScript    const fs = require('fs');  const path = require('path');  const ref = require('ref');    function main() {  // Begin process log  const args = process.argv;    console.log("Running demo...");    let ifile = "B.docx"; // the file to be tagged  let tags = "digitalsignature.xml"; // tags to be used  let tagged\_file = "B-tagged.docx" // file created by GWSecuTag\_RetrieveTagFile()  let retrieved\_tags = "retrieved\_tags.xml" // the tags retrieved from the tagged file  let lib\_file = "gw\_securtag.dll" // the sec tag library file DLL  let wrapper\_script = "Core2JS\_sec\_tag.js"; // the wrapper script    // Include the Glasswall JavaScript Wrapper module  try {  let glasswall = require(wrapper\_script);    // Create an instance of the Glasswall Library  var gw = new glasswall(lib\_file);  }  catch (err) {  console.error("Unable to load Glasswall JavaScript Secure Tagging wrapper");  console.error(err);  console.error("Running demo...failed");  process.exit();  }    Console.log("\n> Calling GWSecuTag\_TagFile...");  rv = gw.GWSecuTag\_TagFile(ifile, tags, tagged\_file);  console.log(`\n> Calling GWSecuTag\_TagFile...done with status ${rv}`);    // call GWSecuTag\_RetrieveTagFile()  Console.log("\n> Calling GWSecuTag\_RetrieveTagFile...");  rv = gw.GWSecuTag\_RetrieveTagFile(tagged\_file, retrieved\_tags);  Console.log(`\n> Calling GWSecuTag\_RetrieveTagFile...done with status ${rv}`);    console.log("Running demo...done");  }    if (require.main === module) {  main();  } |

## Java Wrapper

### Environment

* The Java wrapper requires a JRE to be installed. This wrapper has been tested with V8. Other versions may not be compatible.
* Ensure that a 64-bit version of the JRE is used.
* The Java wrapper requires all supporting files to be stored in the same directory as the wrapper itself. This means that the Core2 libraries and wrapper support files cannot be stored in subdirectories.
* Ensure that the Core2 Libraries and all required folders are accessible to the wrapper.

### Example Code

The following code uses the contents of a specified xml file to tag a given document. This tagged file is then saved in the specified directory.

The tagged file is then inspected, and the tags contained within the file are extracted. This data is then saved in the specified directory.

|  |
| --- |
| class STJavaTestApp  {  public static void main(String[] args)  {  // Load Security Tagging Library & create object  System.loadLibrary("STJavaBridge");  STJavaBridge GWST = new STJavaBridge();    // Tag file  GWST.GWSecuTag\_TagFile("C:\\ST\\Input\\FileToTag.xlsx", "C:\\ST\\tags.xml",  "C:\\ST\\Output\\TaggedFile.xlsx");    // Retrieve tag from file  GWST.GWSecuTag\_RetrieveTagFile("C:\\ST\\Output\\TaggedFile.xlsx",  "C:\\ST\\Output\\RetrievedTag.xml");  }  } |

Python Wrapper

* This has been tested on versions 3.6 and 2.7 of python.

### Environment

* You need to ensure that you have the Glasswall.py wrapper file accessible.
* Ensure that the Core 2 libraries and all required folders are accessible to the wrapper.

Example Code

|  |
| --- |
| from Glasswall import Glasswall  import ctypes as ct     class PythonWrapper:   # create Glasswall object.   gw = Glasswall(r'e:\Core2\_dlls')    # Tag file  gw.tag\_file(r'e:\file\_path', r'e:\data\_to\_tag\_path', r'e:\tagged\_file\_path')    # retrieve tag from file  gw.retrieve\_tag(r'e:\tagged\_file\_path', r'e:\retrived\_tagged\_path') |

C# Wrapper

### Environment

* Ensure that the DLL (“glasswall.core2.csharp.wrapper”) is added as a reference to your environment to be able to use it.
* Ensure that the Core 2 Libraries and all required folders are accessible to the wrapper.

Example Code

|  |
| --- |
| using glasswall.core2.wrapper;  using system;    class Program  {  static void Main(string[] args)  {  // create Glasswall object.  var gw = new Glasswall("e:\\Core2\_dlls");    // Tag file  glasswall.Tag(”e:\file\_path”, “e:\data\_to\_tag\_path”, “e:\tagged\_file\_path”);    // retrieve tag from file  glasswall.RetrieveTag(”e:\ tagged\_file\_path”, “e:\retrived\_tagged\_path”);  }  } |