Pause and auto resume application Documentation

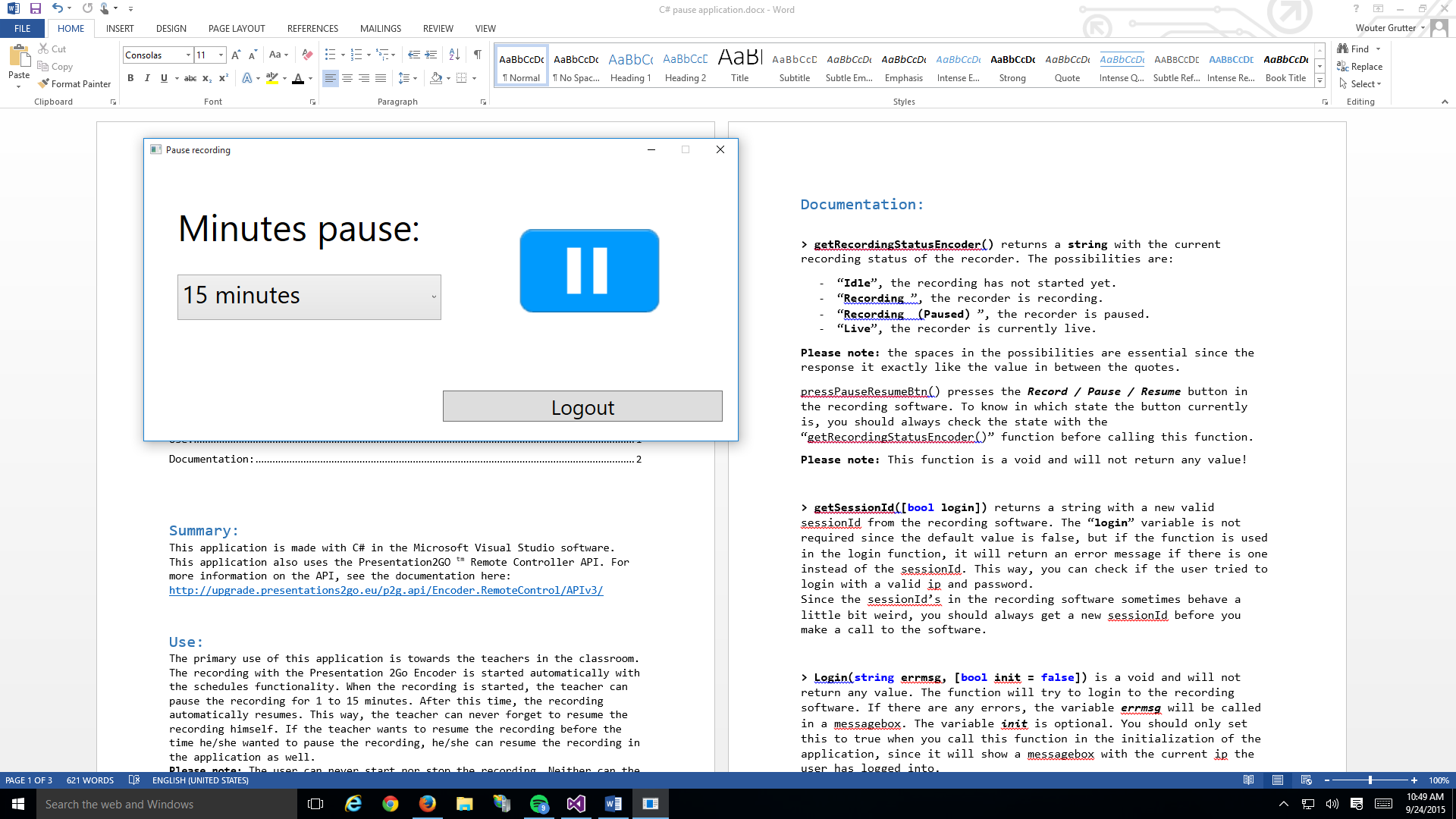
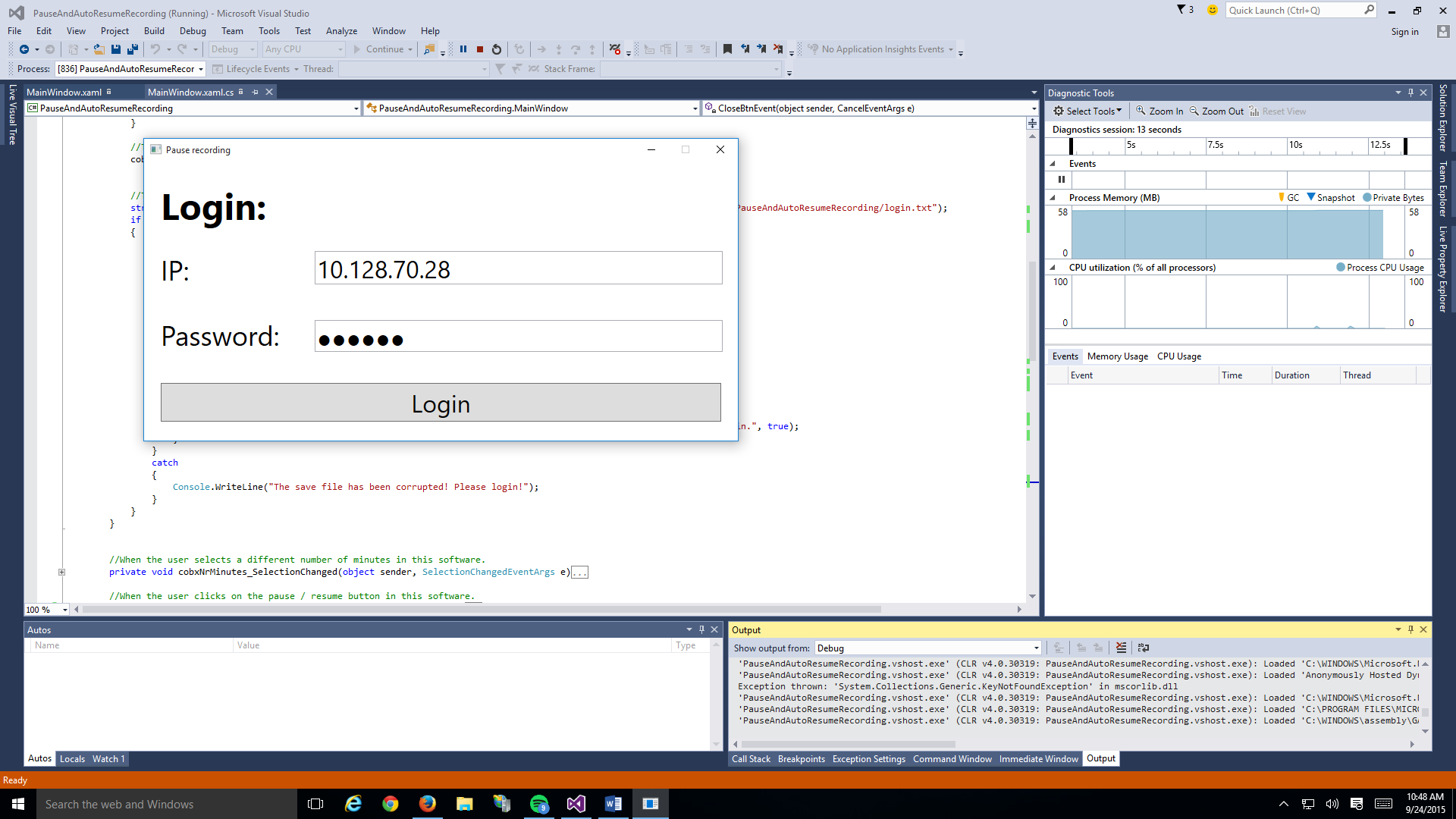


Table of Contents

[Summary: 2](#_Toc430855289)

[Use: 2](#_Toc430855290)

[Documentation: 3](#_Toc430855291)

# Summary:

This application is made with C# in the Microsoft Visual Studio software. This application also uses the Presentation2GO tm Remote Controller API. For more information on the API, see the documentation here: <http://upgrade.presentations2go.eu/p2g.api/Encoder.RemoteControl/APIv3/>

# Use:

The primary use of this application is towards the teachers in the classroom. The recording with the Presentation 2Go Encoder is started automatically with the schedules functionality. When the recording is started, the teacher can pause the recording for 1 to 15 minutes. After this time, the recording automatically resumes. This way, the teacher can never forget to resume the recording himself. If the teacher wants to resume the recording before the time he/she wanted to pause the recording, he/she can resume the recording in the application as well.   
**Please note:** The user can never start nor stop the recording. Neither can the user pause or resume the recording while the recording is live broadcasted.

# Documentation:

**> getRecordingStatusEncoder()** returns a **string** with the current recording status of the recorder. The possibilities are:

* “**Idle**”, the recording has not started yet.
* “**Recording** ”, the recorder is recording.
* “**Recording (Paused)** ”, the recorder is paused.
* “**Live**”, the recorder is currently live.

**Please note:** the spaces in the possibilities are essential since the response it exactly like the value in between the quotes.

pressPauseResumeBtn() presses the ***Record / Pause / Resume*** button in the recording software. To know in which state the button currently is, you should always check the state with the “getRecordingStatusEncoder()” function before calling this function.

**Please note:** This function is a void and will not return any value!

**> getSessionId([bool login])** returns a string with a new valid sessionId from the recording software. The “**login**” variable is not required since the default value is false, but if the function is used in the login function, it will return an error message if there is one instead of the sessionId. This way, you can check if the user tried to login with a valid ip and password.Since the sessionId’s in the recording software sometimes behave a little bit weird, you should always get a new sessionId before you make a call to the software.

**> Login(string errmsg, [bool init = false])** is a void and will not return any value. The function will try to login to the recording software. If there are any errors, the variable ***errmsg*** will be called in a messagebox. The variable ***init*** is optional. You should only set this to true when you call this function in the initialization of the application, since it will show a messagebox with the current ip the user has logged into.

**Please note:** You should always get the login ip and password **BEFORE** you call this function, since it will not try to get these inside the function and will give an error.

**> ChangeVisibility(Visibility visibil)** is a void and will not return any value. This function changes the visibility of the elements in the application. Possible visibility options are:

* Visibility.Hidden : this will hide the login elements and show the control elements.
* Visibility.Visible : this will show the login elements and hide the control elements.

**> WriteToFile(string filePath, string value)** is a void and will not return any value. This function writes the string **value** to the file in **filepath**. If the file does not exist. It will create the file and write the date to the file.

**> ReadFromFile(string filePath)** returns a string of the value inside the file at **filepath**.

**Please note:** If the file does not exist or it has an error reading from the file, it will return **NULL**.

**> Encrypt(string plaintext [, string passPhrase = "YourEncryptionpassword"])** returns a string with the encrypted value of **plaintext**. It will use the password which you specify as the standard password, or you can give it a password you want to use to decrypt the string.

**> Decrypt(string cipherText [, string passPhrase = "YourEncryptionpassword"])** returns a string with the decrypted value of **cipherText** and will decrypt it with the specified password.

**Please note:** the specified password has to be the exact same as the password you encrypted the data with. Otherwise you cannot decrypt the data.