Website: www.chestysoft.com Email: info@chestysoft.com

csIniFile 2.0 - ASP IniFile Editing Component

This component provides a comprehensive set of functions for reading, creating and editing Windows INI files from within an ASP script. It is a COM component and so it can be used in other COM compatible environments, as well as ASP.

The INI file text format was the standard method for Windows 3.x applications to store and retrieve application settings from session to session. An INI file stores information in logical groupings, called "sections". The section name is enclosed in square brackets, for example "[Section1]". Within each section, actual data values are stored in named keys. Keys take the form: <keyname>=<value>

Viewing any INI file with a text editor will reveal this structure.

With the introduction of 32-bit Windows systems, applications typically use the system registry to store and retrieve their settings, instead of INI files. However, there are still many cases where INI files are used effectively. With this component an INI file can easily be used to store the settings of a web application.

A free, fully functional trial version of csIniFile is available. This trial version has a built in expiry date that causes the main functions to stop working after that time. This is the only difference in functionality between the trial and full versions. This means that you can fully test if this component is suitable for your application before considering whether to license the full version.

Version 2.0 is supplied as two different DLL files, one is 32 bit and the other 64 bit. Refer to the next section for more details of registration and component instantiation.

Using These Instructions

These instructions are divided into a number of sections with the relevant methods and properties described in each. There are quick links to some sections below. A full Table of Contents is available on the next page and an index listing all commands in alphabetical order is included at the back for easy reference. The PDF version also has bookmarks for direct navigation to each heading.

Click on one of the links below to go directly to the section of interest:

Chestysoft, October 2013
www.chestysoft.com

TABLE OF CONTENTS

1. Registering the Component and Getting Started

1.1. Registration and Server Permissions

Before the component can be used the DLL file must be registered on the server. This can be done using the command line tool REGSVR32.EXE. Take care to use the correct version of this tool as there is a 64 bit version in the Windows\System32 folder and a 32 bit version in the Windows\SysWOW64 folder. The syntax is:

regsvr32 *dllname*

where *dllname* is the path and name of the DLL to register.

There are two DLL files supplied in the zip archive, one for 32 bit systems and one for 64 bit. The 32 bit file is called csIniFile.dll (csIniFileTrial.dll for the trial version). The 64 bit file is called csIniFile64.dll (csIniFile64Trial.dll for the trial version). The 64 bit file cannot be used on 32 bit systems.

Chestysoft has a free utility that performs the registration function through a Windows interface instead of using regsvr32. This tool can be downloaded from the Chestysoft web site:
www.chestysoft.com/dllregsvr/default.asp

We suggest creating a folder specifically for component DLLs rather than using the Windows System folder as this makes them easier to manage and avoids the naming confusion on the 64 bit systems.

The application that uses the component must have permission to read and execute the DLL. In a web application like ASP this means giving the Internet Guest User account Read and Execute permission on the file. This account must also have the appropriate permissions for file handling. Read permission is required to read/open file from disk. Write permission is required to create a new file and Modify is required to edit or delete an existing file. These permissions can be set in Windows Explorer and applied to either a folder or individual files.

1.2. Object Creation

In any script or programme that uses the component an object instance must be created. The syntax in ASP is as follows.

For the full 32 bit version:

```
Set Obj = Server.CreateObject("csIniFile.Usefile")
```

For the trial 32 bit version:

```
Set Obj = Server.CreateObject("csIniFileTrial.Usefile")
```

For the full 64 bit version:

```
Set Obj = Server.CreateObject("csIniFile64.Usefile")
```

For the trial 64 bit version:

```
Set Obj = Server.CreateObject("csIniFile64Trial.Usefile")
```

In each case the object name is "Obj", but any variable name could be used.

1.3. The Trial Version

The trial version of the component is supplied as a separate DLL called csIniFileTrial.dll (or csIniFile64Trial.dll). This trial version is fully functional but it has an expiry date, after which time it will stop working. The object can still be created after the expiry date but it cannot be used to download files.

The expiry date can be found by reading the *Version* property.

Version - String, read only. This returns the version information and for the trial, the expiry date.

Example:

```
Set Obj = Server.CreateObject("csIniFileTrial.Usefile")
Response.Write Obj.Version
```

Visit the Chestysoft web site for details of how to buy the full version - http://www.chestysoft.com

1.4. Using csIniFile with Component Services

A COM component can be added to a COM+ Application in Component Services. One reason to do this is to be able to run a 32 bit DLL on a 64 bit system. Another is to specify a Windows account to use the component to allow that component to access network files that would be unavailable if the component was called by the default internet guest user.

An online description of configuring Component Services is available here:

http://www.chestysoft.com/component-services.asp

On Windows 2008 and later it is necessary to "Allow intrinsic IIS properties" in the COM+ component properties. csIniFile will run without this but some of the utility functions require IIS intrinsic properties.

1.5. System Requirements

csIniFile version 3.0 does not support earlier Windows operating systems. It requires Windows 2003 or later for a server or Windows XP or later for a desktop. It will not register or run on Windows 2000. We can still provide version 2 for any users of an older operating system.

2. Reading and Writing INI Files

2.1. The FileName Property

Before reading or writing from an ini file, it is necessary to set the *FileName* property to the name and full path of the ini file to used by the component.

FileName - String. Name and physical path to the ini file the object will edit. If the file does not exist, it will be created, if possible.

Example:

```
Obj.FileName = "c:\adirectory\example.ini"
```

2.2. Deleting Entries From the INI File

These two methods delete keys and sections.

DeleteKey *Section*, *KeyName* - Deletes the named key and its value within the named section. **EraseSection** *Section* - Deletes the entire named section.

2.3. Reading INI File Keys

The following read only properties return the value of the named key from the named section. In each case, a third parameter, *Default*, is included which is the value returned if the key is not defined. *Section* and *Keyname* are strings, *Default* is the same type as the property.

ReadString(*Section*, *Keyname*, *Default*) - Returns a string value. **ReadBool**(*Section*, *KeyName*, *Default*) - Returns a Boolean value, True or False. A Boolean value is stored in the ini file as 1 for true, and 0 for false. Any non-zero integer value will read as true. **ReadDate**(*Section*, *KeyName*, *Default*) - Returns a date. **ReadDateTime**(*Section*, *KeyName*, *Default*) - Returns a date and time. **ReadFloat**(*Section*, *KeyName*, *Default*) - Returns a floating point number. **ReadInteger**(*Section*, *KeyName*, *Default*) - Returns an integer number. **ReadTime**(*Section*, *KeyName*, *Default*) - Returns a time.

2.4. Verifying if a Section or Key exists

Two properties are available. SectionExists checks for a named section and ValueExists checks for a named key.

SectionExists(*Section*) - Returns true if the section exists. **ValueExists**(*Section*, *KeyName*) - Returns true if the key exists within the named section.

2.5. Writing Values to the IniFile

For each of the properties listed for reading a value there is a corresponding method to write a value. Attempting to write data to a non-existent section or a non-existent key is not an error. The section and/or key will be created and the new value set.

Each command takes three string parameters, the *Section*, the *KeyName* and the *Value* to write. In VBScript there are no brackets around the parameters. *Section* and *Keyname* are strings, *Value* is the same type as the property.

WriteString *Section*, *KeyName*, *Value*	-	Writes a string value.
WriteBool *Section*, *KeyName*, *Value*	-	Writes a Boolean value.
WriteDate *Section*, *KeyName*, *Value*	-	Writes a date.
WriteDateTime *Section*, *KeyName*, *Value*	-	Writes a DateTime.
WriteFloat *Section*, *KeyName*, *Value*	-	Writes a floating point number.
WriteInteger *Section*, *KeyName*, *Value*	-	Writes an integer number.
WriteTime *Section*, *KeyName*, *Value*	-	Writes a time.

2.6. The Sections and SectionKeys Collections

There are two "collections" provided to read all the sections in an ini file, and all the keys in a section. These can be looped through using the For..Each construct. Both collections have Item and Count properties. Both collections are read only so there is no *Add* method and *Item* cannot be written to.

2.6.1. The Sections Collection

Before reading values from the collection, it is necessary to read the values into it. This takes the form:

```
Sections.ReadSections
```

Example:

```
Obj.Sections.ReadSections
```

This assumes an object called Obj has been created and the *FileName* property has been set. It will fill the *Sections* collection with the names of all the sections in the ini file.

The number of sections is returned by:

```
Obj.Sections.Count
```

The name of the first section is returned by:

```
Obj.Sections.Item(0)
```

Note that it is a zero based array.

2.6.2. The SectionKeys Collection

This must also be initialised before reading from it. This takes the form:

```
SectionKeys.ReadKeys Section
```

Example:

```
Obj.SectionKeys.ReadKeys "Section1"
```

This assumes an object called Obj has been created and the *FileName* property has been set. It will fill the *SectionKeys* collection with the names of all the keys in the section called "Section1".

This example shows how to list all the key names in each section of an ini file.

```
<%
  Set Obj = Server.CreateObject("csIniFile.UseFile")
  Obj.FileName = "c:\adirectory\example.ini"
  Obj.Sections.ReadSections
  For Each Section in Obj.Sections
    Response.Write Section & "<br>"
    Obj.SectionKeys.ReadKeys Section
    For Each Key in Obj.SectionKeys
      Response.Write "---" & Key & "<br>"
    Next
  Next
%>
```

3. File Utilities

There are a number of file utility functions included for convenience. They are not intended to be a comprehensive set, because standard ASP has the File System Object to cover most file utilities

CurrentDir - This property returns the actual path of the directory containing the script. It is complete with the trailing backslash character. It only works with ASP.

ParentDir(*Directory*) - *Directory* is a string value and must be a full directory path. The return value is the parent directory.

Example:

```
Response.Write Obj.ParentDir(Download.CurrentDir)
```

This would display the parent directory to the one containing the current script.

ScriptName - A read only property returning the current script name complete with extension. This only works with ASP.

FileSize(*FileName*) - *FileName* is the full path and filename of a file. The return value is the file size in bytes.

Delete(*FileName*) - This deletes the file *FileName*. Note that it is permanently deleted, NOT placed in the Recycle Bin.

Copy *OldName*, *NewName* - This copies the file *OldName* to the location and name given by *NewName*. Full paths are required.

Rename *OldName*, *NewName* - This renames the file *OldName* to *NewName*. Full paths are required, and so renaming to a different directory is the equivalent of moving the file.

AppendToFile *FileName*, *NewLine* - This appends the string *NewLine* to the text file *FileName*. If the text file does not exist, it will be created if possible. The full physical path is required.

Example:

```
Obj.AppendToFile Obj.CurrentDir & "test.txt", "Hello"
```

This will append the line "Hello" at the end of a text file called test.txt which is in the same directory as the current script. If the file does not exist it will create it.

AppendToFile is the only command in this component for manipulating text files. It is useful for maintaining a simple log file or for recording debug information. There is a full set of commands for dealing with text files in the built in File System Object.

All the file handling routines require that the Internet Guest Account has the appropriate permissions on the server, otherwise errors will result.

4. Revision History

The current version of csIniFile is 2.0.

New in Version 2.0

64 bit version released.

5. Other Products From Chestysoft

Visit the Chestysoft web site for details of other COM objects.

ActiveX Controls

csXImage	-	ActiveX control to display, edit and scan images.
csXGraph	-	ActiveX control to draw pie charts, bar charts and line graphs.
csXThumbUpload	-	Upload multiple files by HTTP or FTP with previews and image edits.
csXPostUpload		Uploads batches of files from a client to a server using an HTTP post.
csXMultiUpload	-	Select and upload multiple files and post to a server using HTTP.

ASP Components

csImageFile	-	Resize, create and edit images.
csDrawGraph	-	Draw pie charts, bar charts and line graphs in ASP.
csASPGif	-	Create and edit animated GIFs.
csFileDownload	-	Control file downloads from an ASP script.
csASPUpload	-	Process file uploads through a browser.
csASPZipFile	-	Create zip files and control binary downloads.
csFTPQuick	-	ASP component to transfer files using FTP.

ASP.NET

csASPNetGraph	-	.NET component to draw pie charts, bar charts and line graphs.
csNetUpload	-	ASP.NET component for saving HTTP uploads.
csNetDownload	-	ASP.NET class to control file downloads.

Web Hosting

We can offer ASP enabled web hosting with our components installed. Click for more details.

6. Alphabetical List of Commands