



CurtainTM e-locker 3.1

Curtain Plug-in SDK

Prepared by: Coworkshop Solutions Ltd.
Suite D, 7/F, West Gate Tower
7 Wing Hong Street
Lai Chi Kok, Kowloon

Version: 2.3

Date: November 21, 2005

Contact your Authorized Curtain e-locker Reseller or Service Provider to report problems and/or provide feedback.

Additional help resources or updates will be available by emailing info@coworkshop.com

Coworkshop Solutions Ltd. reserves the right to make changes to this document and to the product described herein without notice. The software described in this manual is furnished under the terms and conditions of the Curtain e-locker Software License Agreement and may be used or copied only in accordance with the terms of the agreement.

For information about your legal rights concerning the use of the Curtain e-locker, please refer to the Curtain e-locker Software License agreement.

© 2002-2005 Coworkshop Solutions Ltd. All Rights Reserved. Curtain belongs to Coworkshop Solutions Ltd. All other brand names, product names, or trademarks belong to their respective holders.

Table of Content

1. Curtain™ e-locker	4
2. Curtain™ e-locker Plug-in SDK	5
3. Using Curtain e-locker in Script.....	6
3.1 Authorization String	6
3.2 Curtain Plug-in Component.....	6
3.3 Authorization for Uploading Files	7
3.4 Launching the Controlled Application Program	8
4. Curtain Plug-in Reference	9
Curtain Plug-in Component.....	9
Curtain.PlugIn	9
Methods	10
Authorize Method.....	10
AccessDone Method.....	11
StartMonitor Method.....	12
AuthorizeUpload Method.....	13
GetProtectedPath Method.....	14
GetOtherPath Method.....	15
5. Integrating Curtain™ e-locker with Microsoft asp.net application	16
5.1 Introduction	16
5.2 Web form: files.aspx	16
5.3 Web form: downloadFile.aspx	16
5.4 Web form: uploadFile.aspx	17
5.5 Encryption and Encryption key	17
5.6 Curtain ActiveX Control and script.js.....	17
6. Getting Help	19
6.1 Get Help	19
6.2 Get Technical Support.....	19

1. Curtain™ e-locker

As in the Information Age, organizations are well aware of the downside of information sharing within the corporation especially when the data files are highly sensitive, for example, customer information, product design, proposal, financial report and contract. Workstations are usually lightly guarded even the sensitive files are downloaded to them. People now have easy access to the workstations and the sensitive files even they do not have the security clearance or authority to do so. How to prevent files from being copied out of the workstations and leak out of organization? Policies and procedures are not good enough to protect your information ownership; Curtain e-locker is the solution to fix this problem.

Curtain e-locker is a File Rights Management system that prevents your sensitive files from leaking out of the organization. Curtain e-locker allows users to access sensitive files by using the native software (not through a third party file viewer), such as MS Word, MS Excel, AutoCAD and SolidWorks, but NOT to print, save as, or send files to external sources, such as floppy disks, USB storage devices, CD-R/RW, and Internet mail.

2. Curtain™ e-locker Plug-in SDK

This SDK contains information on how to use Curtain e-locker 3.1 in script programming level. This enables certain application programs can make use of script to access files in protected directory. This feature of Curtain e-locker is mainly used by Database programs to put files into protected directory or retrieve files from protected directory and put back to the database.

3. Using Curtain e-locker in Script

The main purpose of using Curtain e-locker in script is to let application program that running the script can access files in protected directory. Therefore, it should be very careful that only the necessary application program should use Curtain e-locker in script.

3.1 Authorization String

When script calls Curtain Plug-in to request the Plug-in to authorize the script program to access Protected Directory, an *Authorization String* must be given by the script.

Authorization String is a special string provided by the reseller or distributor of Curtain e-locker when product activation of Curtain e-locker is done. System administrators or script programmers must put the Authorization String as a parameter to the calls to Curtain Plug-in. The Curtain Plug-in uses this Authorization String to verify whether the caller is trusted or not and to determine whether to authorize the caller to access the protected directory or not.

3.2 Curtain Plug-in Component

The only component that script should call is the **Curtain Plug-in**, which is used to acquire the authorization from Curtain e-locker to access files stored in the protected directory.

There are 3 steps when using the Plug-in component.

1. Acquire authorization from Curtain e-locker.
2. Access files in the protected directory.
3. Notify Curtain e-locker that operations on the files in the protected directory are done.

The following is an example of using Curtain Plug-in component:

```
Set oAuth = CreateObject("Curtain.PlugIn")

Dim sProtDir As String
Dim sAccPath As String
sProtDir = oAuth.Authorize("Xhq4ye-qzaaic-cqzhc4-4XXa4q-ieyzia-h44chX-
aya4Xc-ychzcX-hi", sFileName, 1)
if sProtDir = ""
    then MessageBox This client is not authorized to use the Curtain
system!", MB_OK + MB_ICONWARNING, "Error"
    Exit Sub
End if
sAccPath = sProtDir + "\" + sFileName
```

'Access the file by the path sAccPath

Call oAuth.AccessDone

In the above code segment, it first creates an instance of the Curtain Plug-in component 'Curtain.PlugIn' and assigns it to the variable **oAuth**. Then, it performs the step 1 by calling the **Authorize** method of the Plug-in component. The first parameter of the Authorize method is the Authorization String and the second parameter is the filename of the file that will be accessed. The last parameter is an integer to indicate whether the file must be protected. If this is non-zero, the file will be protected and authorization will be done regardless of the extension of the filename.

If the authorization is completed successfully, the path to the protected directory will be returned. The full path **sAccPath** to the file that will be accessed is constructed. Otherwise, if the authorization is failed, empty string is returned. In this case, the files in the protected directory cannot be accessed in the script.

The step 2 is accessing the file by using the full path **sAccPath**. At this moment, the script is authorized to access the files in the protected directory. The script can put new files to the protected directory or read the files in the protected directory.

Finally, in the step 3, the script calls the method AccessDone to tell the Plug-in component that the access to the protected directory is done. After this call, the script cannot access the files in the protected directory any more.

3.3 Authorization for Uploading Files

If your system requires users to specify a file in the protected directory in order to upload the file, you should use **AuthorizeUpload** method to allow users to locate the file in Protected Directory first. And then, you can use **Authorize** method to upload the file to your system (the authorization described in Section 3.2).

Curtain Plug-in component provides a simple method **AuthorizeUpload** for users to specify a file in the protected directory. The script can have the following statement to let users to specify a file in the protected directory:

```
Dim sLocalPath As String
sLocalPath = oAuth.AuthorizeUpload("Xhq4ye-qzaaic-cqzhc4-4XXa4q-ieyzia-
h44chX-aya4Xc-ychzcX-hi")
if sLocalPath = "" then
    Exit Sub
End if
```

The parameter in **AuthorizeUpload** method is the same Authorization String to the **Authorize** method. When **AuthorizeUpload** method is called in script, a controlled file browser dialog box will show files in the protected directory. Users can select a file from the dialog. Then, the full path of the selected file will be returned, and the script can use this full path to access the file from the protected directory and upload it to the database server. Finally, the script must call **AccessDone** method to complete the access of the file.

3.4 Launching the Controlled Application Program

The Plug-in component has one more function, it can open a file in protected directory with protected application. The protected application program can be started by using the StartMonitor method.

```
Call oAuth.StartMonitor(sAccPath)
```

The **sAccPath** variable is the full path of the file in the protected directory that will be opened by the protected application program. If file specified is a file type of certain protected application program, the protected application will be started automatically. This method should be called between the Authorize method and the AccessDone method.

4. Curtain Plug-in Reference

This reference manual describes the Plug-in component and the methods of the component in details.

Curtain Plug-in Component

Curtain.PlugIn

The **Curtain.PlugIn** component provides authentication service to the client program that needs to use Curtain plug-in to access the files in the protected directory. Whenever an application program needs to access files in the protected directory, it must use a script to call the Curtain.PlugIn component to request authorization to access the protected directory. Otherwise, the application program cannot access the files in the protected directory.

When using the PlugIn component, the script must first create an instance of the component, and then call the **Authorize** or the **AuthorizeUpload** method to request authorization. Then, the script can access files in the protected directory if authorization is completed successfully. After the access of files is done, the script must call the **AccessDone** method to tell the component to end the access. If script needs to launch the protected application program to open files in protected directory, the script can call **StartMonitor** method of the component to start a protected application.

Methods

Authorize Method

Syntax:

```
Authorize( sAuthorizeString, sFileName, lToProtect )
```

Parameters:

sAuthorizeString

This is a string provided by the reseller or distributor of Curtain e-locker during the process of Product Activation. This string contains information of the machine which has been installed Curtain Central Administrator, so it is different for different customers and it will be invalid when it is used in different machine. Please refer to Section 3.1 for more information.

sFileName

This is the filename of the file that will be accessed. In fact, the Plug-in component uses the extension of this filename to determine whether the file should be in protected directory or not. This filename will not be used for any other purpose.

lToProtect

This is an integer to indicate whether the file should be resides in protected directory or not. If it is zero, the Plug-In component uses the extension of the *sFileName* parameter to determine the protection. If it is non-zero, the file will be treated as protected regardless the extension of the *sFileName* parameter.

Description:

This method requests Curtain to authorize the process that calling this method to access files in the protected directory. If authorization is successfully acquired, this method returns the path to the protected directory; otherwise, empty string is returned. Any program that needs to access files in the protected directory must call this method first before it can access the files, otherwise the access will be denied. When program finished the access of the files, it must call the **AccessDone** method to tell Curtain that accessing to the files is done.

In order to safety protect files in protected directory, the *sAuthorizeString* parameter must be kept in secret. If this string is get known to unauthorized person, he can use this string to access files in the protected directory.

AccessDone Method

Syntax:

AccessDone

Description:

This method notifies Curtain that access of files in the protected directory is done. After this method is called, the program calling this method cannot access files in the protected directory any more. In order to protect files in the protected directory, the program that called the **Authorize** or **AuthorizeUpload** methods to access the files must call this method after all necessary access operations.

StartMonitor Method

Syntax:

```
StartMonitor( sDataFilePath )
```

Parameters:

sDataFilePath

This is the full path of the file which will be opened by a protected application. This method will start the application program according to the extension of the file specified.

Description:

This method opens the file specified with a protected application program according to the extension of the file. However, if the file is not recognized as any protected file type, this method will call the OS shell to open the file, in this case, the application program cannot access the file.

This method should be called in between **Authorize** method and **AccessDone** method.

AuthorizeUpload Method

Syntax:

```
AuthorizeUpload( sAuthorizeString )
```

Parameters:

sAuthorizeString

This is a string provided by the reseller or distributor of Curtain e-locker during the process of Product Activation. This string contains information of the machine which has been installed Curtain Central Administrator, so it is different for different customers and it will be invalid when it is used in different machine. Please refer to Section 3.1 for more information.

Description:

This method shows a controlled file browser dialog box for user to select a file in the protected directory, and then it requests the authorization from Curtain in order to access the files in the protected directory. Finally, it returns the full path to the file that the user selected. It returns empty string if user cancels the selection or the authorization is not succeeding.

If this method is done successfully, the program that calling this method gets the full path to a file (that is selected by user) in the protected directory and it can access that file. In normal case, the program should read that file and upload it to the database server. After the access is done, the program must call the **AccessDone** method to tell Curtain the end of the access.

GetProtectedPath Method

Syntax:

```
GetProtectedPath()
```

Description:

This method simply returns the path of the protected directory to the caller. It will return empty string if there is any error in retrieving the path of the protected directory.

GetOtherPath Method

Syntax:

```
GetOtherPath()
```

Description:

This method returns a path of the directory that non-protected files will be downloaded to. Whenever the **Authorize** method found that the requested file is not a protected file, it will return this path and will not authorize the caller to access the protected directory.

5. Integrating Curtain™ e-locker with Microsoft asp.net application

5.1 Introduction

We provide a sample asp.net application. With the sample, Asp.net Developers would find integration of their web application and Curtain e-locker an easy job.

The sample basically consists of a set of libraries written in c#, a javascript file ,which is running in browser, and 3 Asp.net web forms.

To compile the sample, please install Microsoft.Net platform 1.1 or above and create a virtual directory named **CurtainSample** in the Microsoft IIS server.

5.2 Web form: files.aspx

The web form, `Files.aspx`, is a central place for end users to browse, download and upload files. First of all, the web form shows a list of files under the folder `c:\temp`.

Download a file

When a user clicks on those links, the browser creates an ActiveX control instance. The ActiveX control is responsible to (1) get an encrypted stream of the entitled file from other web form, namely `downloadFile.aspx`, (2) decrypt the stream and (3) save file to the directory.

Upload a file

When the user clicks on the upload link, the browser creates an ActiveX control instance and triggers the upload process. The ActiveX control will (1) freeze the browser, (2) open a file selection dialog, (3) decrypt the selected file and (4) post the decrypted stream to a web form, namely `uploadFile.aspx`.

5.3 Web form: downloadFile.aspx

When the Curtain ActiveX control, (or a hacker who tries to get a file) request the web form, `downloadFile.aspx`, the web form examines the http headers which contains the information about the selected path and filename. (Although the fields are originally designated for path and filename, asp.net developers can make use of the headers to identify the select file by their own scheme.) With the headers and some kind of developer defined conversion, the web form identifies which file the requester wants to download. It reads and encrypts the selected file and response to the requester an encrypted stream. The stream is encrypted by strong key and algorithm. Nobody is able to steal the file by

getting the stream from the web form or even by some kind of network analysis tools.

Additional to the file stream, a MD5 hash code of the file will send to the requester for verification purpose.

5.4 Web form: uploadFile.aspx

Likewise, the web form, `downloadFile.aspx`, receives the encrypted stream, decrypts and saves the file to the selected location in server. With a similar developer defined conversion, the web form makes use of the HTTP headers to identify the upload location in server.

The web form checks the file integrity and the identity of upload by the hash code of the file and the hash code the encryption key.

5.5 Encryption and Encryption key

The encryption algorithm in the processes of upload and download are 128 bit AES. The AES encryption key is encoded and as a text file, namely `authorStr.txt`, and by default stored in the folder, `c:\inetpub`, of the web server. The file is generated during the product activation process. Within the 30 days evaluation period, the system does not need the file and use a default encryption key.

5.6 Curtain ActiveX Control and script.js

An ActiveX Control is running in the web browser. It has 2 main functions. 1) It accesses the Curtain protected directory on behalf of the web browser. 2) It encrypts and decrypts the file stream during uploading and downloading. Without the Curtain ActiveX Control, the browser can't access the protected directory and can't decrypt the ciphered file.

The COM object has a Program ID

```
Curtain.ClientPlugIn
```

The interface of the COM object is simple.

```
BSTR SPCOMMAND(BSTR sCommand, BSTR sURL, BSTR sPath, BSTR  
sFilename);
```

sCommand

The value of `sCommand` is either **SPDownload** or **SPUpload** for downloading and uploading respectively.

sURL

The value is the URL of the web form for downloading and uploading.

sPath / sFilename

The values are the path and the filename of the target file. Asp.net developers can make uses of the two fields to identify file in their own scheme. The values are encoded in the curtain HTTP headers. The web form can get the exact values. For example, an Asp.net web application stores documents in SQL server. The developers can pass primary keys to identify the files.

Please refer to the javascript file, script.js, in the sample asp.net application. It demonstrates how to use the ActiveX control to download and upload files.

6. Getting Help

6.1 Get Help

Online help provides information and instructions for using Curtain e-locker to protect your sensitive information.

To access online help while you work:

1. Launch Curtain e-locker Client.
2. Select "Help" -> "Curtain Client Help" in menu. Then, Curtain e-locker Client Guide will be shown.

6.2 Get Technical Support

For technical support of Curtain e-locker, you can contact your Curtain e-locker service provider. For further support, you can:

- Contact authorized Curtain e-locker distributors in your region. You can locate the information in our website, <http://www.coworkshop.com>.
- Send email to info@coworkshop.com. Please specify your existing version of Curtain e-locker system with detailed description of your question. We will get back to you as soon as possible.