CAS/Tester

Automated Code Access Security Testing for .NET

Version 1

for Visual Studio .NET

by

Desaware, Inc.

Rev 1.0.0 (05/03)

Desaware, Inc.
Software License

Please read this agreement. If you do not agree to the terms of this license, promptly return the product and all accompanying items to the place from which you obtained them.

This software is protected by United States copyright laws and international treaty provisions.

This program will be licensed to you for use only on a single computer. If you wish to install it on additional computers, you must purchase additional software licenses. You may (and should) make archival copies of the software for backup purposes.

You may not make copies of this software for other people. Companies or schools interested in multiple copy licenses or site licenses should contact Desaware, Inc. directly at (408) 377-4770.

You have a royalty-free right to incorporate any of the sample code provided into your own applications with the stipulation that you agree that Desaware, Inc. has no warranty, obligation or liability, real or implied, for its performance.

Licensing: CAS/Tester uses the Desaware Licensing System Component. This framework provides for the transfer of licensing information from the system upon which CAS/Tester is installed, to Desaware's Licensing Web Service. This in turn creates and activates a license key that allows you to use CAS/Tester on your system. The licensing information transferred is a one way cryptographic hash that does not include any personal information, or information that could be used to identify the originating system.

File Descriptions: You may not distribute any compiled files included with CAS/Tester. The files prohibited are: Desaware.CASTester.exe, Desaware.CASTester11.exe, Desaware.MachineLicense.dll, Desaware.MachineLicense11.dll, Desaware.CASTester.dll, Desaware.CASTester11.dll, CASTesterAddIn.dll, CASTesterAddIn11.dll, CASTesterAddIn.tlb, CASTesterAddIn11.tlb, CASTesterInstaller.exe, CASTesterInstaller11.exe, CASTesterLauncher.exe, CASTesterLauncher11.exe, CASTesterUI.dll, CASTesterUI11.dll, and CASTester10.dlsc.

Source Files: Source code for portions of CAS/Tester are included for educational purposes only. You may use this source code in your own applications only if they provide primary and significant functionality beyond that included in the software product. You may not use this source code to develop or distribute components and tools that provide functionality similar to all or part of the functionality provided by any of the components or tools included in the CAS/Tester package.

Limited Warranty

Desaware, Inc. warrants the physical CD and physical documentation enclosed herein to be free of defects in materials and workmanship for a period of sixty days from the date of purchase.

The entire and exclusive liability and remedy for breach of this Limited Warranty shall be limited to replacement of defective CD(s) or documentation and shall not include or extend to any claim for or right to recover any other damages, including but not limited to, loss of profit, data or use of the software, or special, incidental or consequential damages or other similar claims, even if Desaware, Inc. has been specifically advised of the possibility of such damages. In no event will Desaware, Inc.'s liability for any damages to you or any other person ever exceed the suggested list price or actual price paid for the license to use the software, regardless of any form of the claim.

DESAWARE, INC. SPECIFICALLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Specifically, Desaware, Inc. makes no representation or warranty that the software is fit for any particular purpose and any implied warranty of merchantability is limited to the sixty-day duration of the Limited Warranty covering the physical CD and documentation only (not the software) and is otherwise expressly and specifically disclaimed.

This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

This License and Limited Warranty shall be construed, interpreted and governed by the laws of the State of California, and any action hereunder shall be brought only in California. If any provision is found void, invalid or unenforceable it will not affect the validity of the balance of this License and Limited Warranty, which shall remain valid and enforceable according to its terms.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, as applicable. Contractor/Manufacturer is Desaware, Inc., 1100 East Hamilton Avenue, Suite 4, Campbell, California 95008.

Table of Contents

Introduction

What if it were possible for an end user or system administration to arbitrarily prevent your application from performing critical tasks, such as reading or writing files, accessing the registry, or displaying certain types of Windows?

Welcome to the world of .NET.

Code Access Security allows client systems to restrict access to a wide variety of system resources based on where your application originated, who developed it, or any other criteria the client chooses. If you plan to distribute .NET assemblies, whether they are controls, class libraries or applications, you need to know what your assembly will do under a wide variety of security configurations. You need to be sure you are detecting security errors and handling them, disabling features as needed and/or reporting or logging appropriate errors to the client.

CAS/Tester makes it easy to quickly test an assembly under a virtually unlimited number of security configurations, and allows you to disable permissions one at a time to isolate exactly how a specific security setting will impact your assembly.

You do not need an in-depth understanding of Code Access Security to use CAS/Tester – when you run CAS/Tester on your assembly you will, by default, run over eighty security tests on your assembly. However, you will need to understand Code Access Security to use scripting to define your own tests and test scripts.

How CAS/Tester Works

CAS/Tester is an Application Domain host. This means that it, like ASP.Net or the .NET runtime, is able to host Application Domains. CAS/Tester creates a new Application Domain for each test, ensuring that the side effects of one test can't interfere with the results of other tests.

For each test, CAS/Tester creates the requested security configuration, then performs a number of tests:

If the assembly is an **executable (.exe)** file:	CAS/Tester attempts to run the executable under the specified security setting.
If it is a **DLL**:	CAS/Tester loads the assembly.
If it is a **DLL** containing a **User Control**:	CAS/Tester retrieves its own form to act as a container for any controls that you create.
If an **object** is **specified**:	CAS/Tester attempts to create the requested object under the specified security setting.
If the **object** has a **constructor**:	the constructor will execute.
If a **method** is **specified**	CAS/Tester attempts to invoke the method under the specified security setting. The method should take no parameters.

After the test is complete, CAS/Tester attempts to terminate and unload the application domain, and then continue to the next test. CAS/Tester will do its best to close any windows and terminate the application. If CAS/Tester is unable to terminate the application within a specific amount of time, an error will be reported and testing will end.

It is also possible to specify a test script. Test scripts can define security settings, and include any test code you wish to execute.

CAS/Tester handles security settings, threading, and many other tasks. However, it is up to you to specify the method to call or the test script to run. You, the developer also determines the coverage of the test.

CAS/Tester is purely a runtime test – it does no design time analysis of your code.

Using CAS/Tester

Two versions of CAS/Tester are included, one for use with Visual Studio .NET 1.0 (or 2002), and one for use with Visual Studio .NET 1.1 (or 2003). All CAS/Tester files compiled with Visual Studio .NET 1.1 have "11" appended to the file name. You can install each version independently. You should use the CAS/Tester version that corresponds to the framework version used by the assembly that you are testing.

CAS/Tester can be run as a command line utility, or used with two available user interfaces: an add-in that runs within Visual Studio and a standalone application.

Using CAS/Tester From the Command Line

The command line for CAS/Tester is as follows. Path names that include spaces should be bracketed by double quotes.

Desaware.Castester.exe *testassembly* [*object*[*.method*] | *scriptfile.vb*|*.cs*] [*options*]

testassembly is the name of the assembly to test. The path is relative to the executable directory (or you can specify the full path to the assembly). You should include the extension .dll or .exe.

object is an optional name of an object to create in the assembly. If a method is specified, the object is created and the specified method called. The method should have no parameters.

Instead of an object and method, you can specify a script file. If a .vb or .cs extension is specified, CAS/Tester assumes you wish to run a script. All security settings and tests performed will be as specified by the script file.

Additional CAS/Tester options will be described shortly.

Using CAS/Tester from Visual Studio (Add-in Version)

The CAS/Tester Launcher utility is integrated into Visual Studio .NET as an Add-in. This is done during installation by the CAS/Tester Launcher.

You can run the CAS/Tester Launcher by selecting the Visual Studio .NET **Tools - CAS/Tester Launcher** menu.

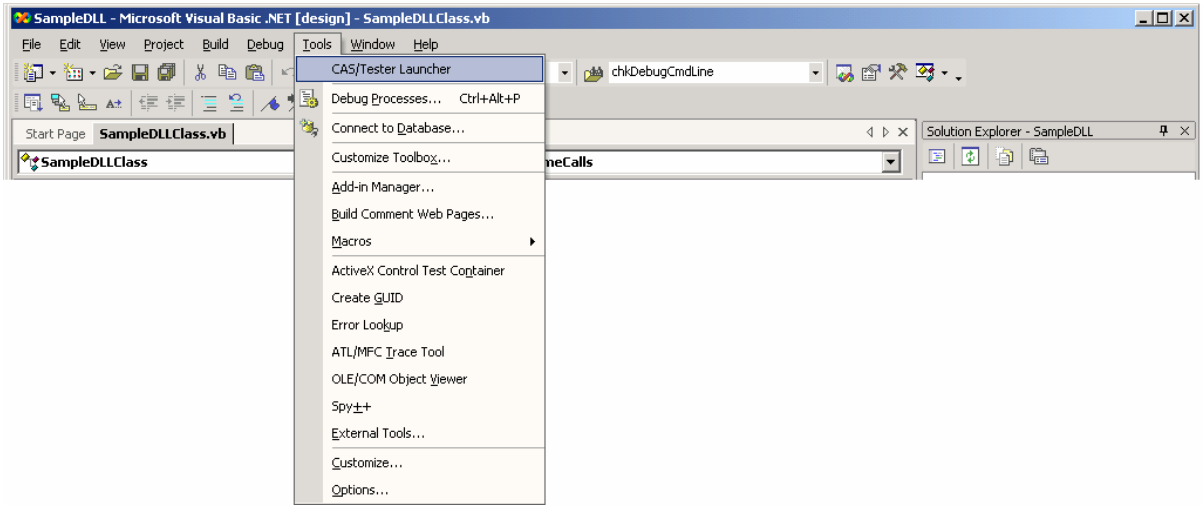

Figure 1
CAS/Tester Launcher Add-In

Initially, the **CAS/Tester Launcher** menu item is automatically added to the **Tools** menu each time you open Visual Studio. You may change this behavior so that the **CAS/Tester Launcher** menu item is manually added to the **Tools** menu as needed by using the Visual Studio .NET Add-in Manager.

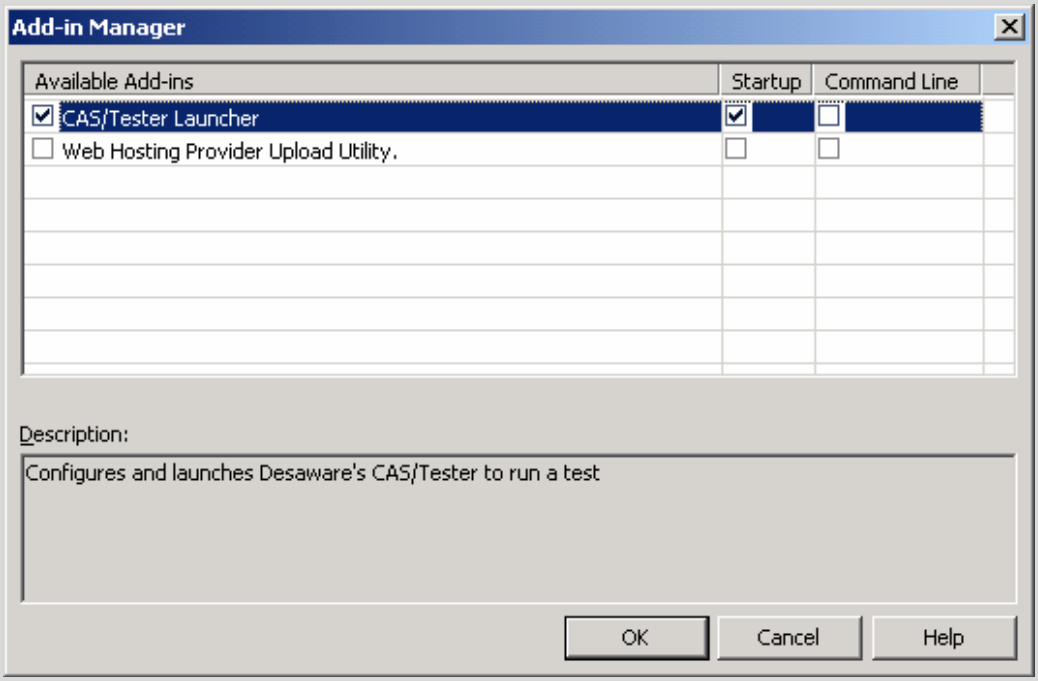

Figure 2
Visual Studio Add-In Manager

Select the **Tools – Add-in Manager** menu. From the Add-in Manager's form, select or unselect the **Startup** check box to specify whether the **CAS/Tester Launcher** menu item is automatically added to the **Tools** menu each time you open Visual Studio. Select or unselect the **Available Add-ins** check box to immediately add or remove the **CAS/Tester Launcher** menu item from the **Tools** menu.

The CAS/Tester Launcher runs as a modeless form on top of other forms in the Visual Studio .NET IDE. You can run successive tests using CAS/Tester Launcher and view the test results without having to close CAS/Tester Launcher after each test. You can also minimize the CAS/Tester Launcher form to access other windows in the Visual Studio .NET IDE. If the Visual Studio .NET IDE is closed or minimized, CAS/Tester Launcher will also be closed or minimized.

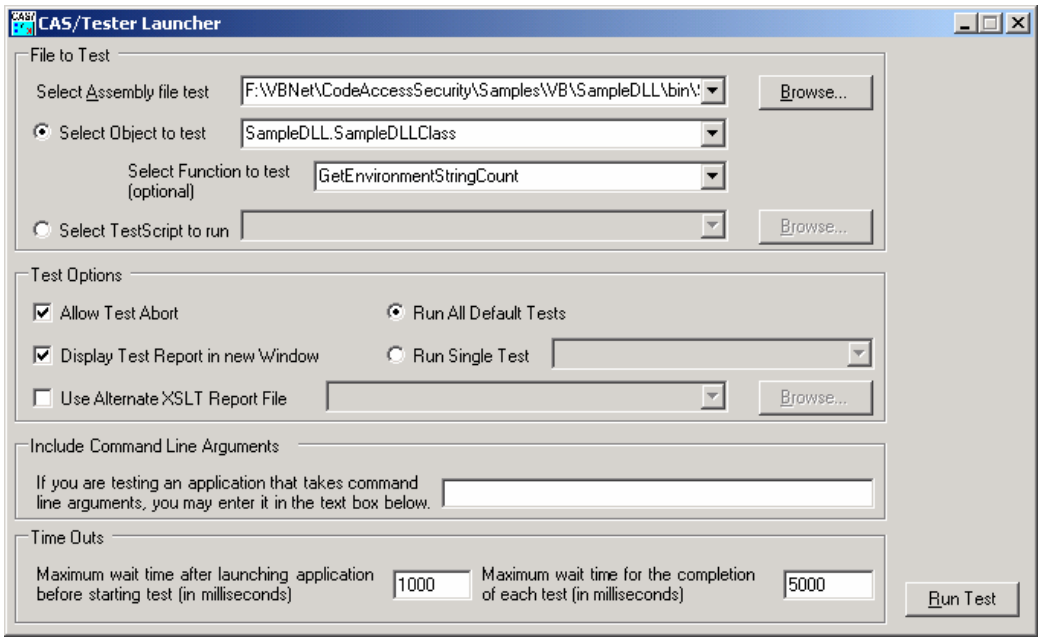

Figure 3
CAS/Tester Launcher

The top frame entitled **File to Test** in the CAS/Tester Launcher contains information about the test. Select the name of the assembly to test from the **Select Assembly file test** combo list box. This combo list box is initially filled with the assemblies of project files for the current solution and the most recently tested assembly files. You may also select other assemblies using the **Browse** button next to the **Select Assembly file test** combo list box.

Optionally, you may select a specific object from the selected assembly to test by selecting the **Select Object to test** option button, then selecting the object from the **Select Object** combo list box. This combo list box is filled with the objects of the selected assembly to test.

You may also select a specific function of the selected object to test. If a function is specified in the **Select Function to test** combo list box, the object is created and the specified function called. The function should have no parameters.

You can also specify a script file to run by selecting the **Select TestScript to run** option button, then selecting a script file from the TestScript combo list box or by using the **Browse** button next to the TestScript combo list box to select a script file. The TestScript combo list box is filled with a list of the most recently tested script files.

The **Test Options** frame allows to you specify additional choices for the test. Select the **Allow Test Abort** check box to have CAS/Tester display an Abort form while running the tests. This allows you to abort the test sequence between tests. Note that the current test will execute to completion. Select the **Display Test Report in new Window** check box to display the test report in a new window pane inside the Visual Studio .NET IDE. If this check box is not selected, each new test report displayed will close the previous report.

Select the **Use Alternate XSLT Report File** to override the default XSLT Transformation file used by CAS/Tester to display the test results. If you select this option, you should select a valid XSLT file in the **XSLT Report File** combo list box, or use the corresponding **Browse** button to specify a XSLT file. Select the **Run All Default Tests** option button to have CAS/Tester test the assembly using all of the pre-defined default permission sets. Select the **Run Single Test** option button and a test name from the accompanying combo list box to have CAS/Tester test the assembly using just the specified permissions. Each of the pre-defined test names are described further by a ***Tool Tip*** when that particular test is selected.

In addition to the standard pre-defined tests, this combo box is also loaded with the standard permission sets and every other permission set found at the Machine level of the test system. Note that when used with a Test Script file, the single test option selected must be defined in the Test Script. When a Test Script file is used and the Run All Default Tests option is selected, then only the permission sets defined in the Test Script file will be used.

For details of the permissions denied or allowed for each test, please search the ClassTemplate.* template file for the name of the test and locate the corresponding constant reference for that name.

The **Include Command Line Arguments** frame allows you to pass a command line argument to your assembly when tested by CAS/Tester. This is commonly used when testing Console applications or Windows Forms applications. Enter the command line string to use in the text box.

The **Timeouts** frame contains two text boxes allowing you to specify timeouts for the CAS/Tester test.

Select the **Run Test** button to launch CAS/Tester with the specified test options. The CAS/Tester Launcher will verify the options and launch CAS/Tester. Upon completion of the test, the test report will be displayed in a Visual Studio IDE window panel. You may leave the CAS/Tester Launcher running to check the test results before running the next test. The CAS/Tester Launcher will automatically minimize when the test is run. The CAS/Tester Launcher will also copy the CAS/Tester command line used for the current test to the system clipboard. This may be useful if you wish to create a batch file of tests.

When you close the CAS/Tester Launcher, it saves the most recent test options selected to an initialization file. The next time the CAS/Tester Launcher is opened, it initializes the majority (the exceptions being those that must be input for each test) of the option fields to the previous settings.

Using the CAS/Tester Launcher as a Standalone Application

The CAS/Tester Launcher utility can also be run as a standalone application. The CAS/Tester installation installs a shortcut menu to the CAS/Tester Launcher. You can start the CAS/Tester Launcher utility by selecting the CAS/Tester Launcher menu item or by running the CASTesterLauncher.exe file located in CAS/Tester's "bin" folder.

Running the CAS/Tester Launcher utility as a standalone application is similar to running it within the Visual Studio environment. Refer to the previous section for the descriptions of the various CAS/Tester Launcher controls. When running as a standalone application, the following items are different than when running as an Add-in:

The **Select Assembly file test** combo list box contains only the most recently tested assembly files. The **Display Test Report in new Window** check box is not visible.

CAS/Tester Tests

By default, CAS/Tester runs over eighty tests on your assembly. A listing of these tests is located in Appendix A of this manual. These include tests denying a variety of standard permissions, denying access to a selection of common file directories and the registry, and various other permissions. It also runs tests using the standard permission sets, and using every other permission set found at the Machine level of the test system.

With scripting, you can define additional tests using any criteria you choose.

Each test has a one word test name and a description.

CAS/Tester Reports

The CAS/Tester security report consists of five sections.

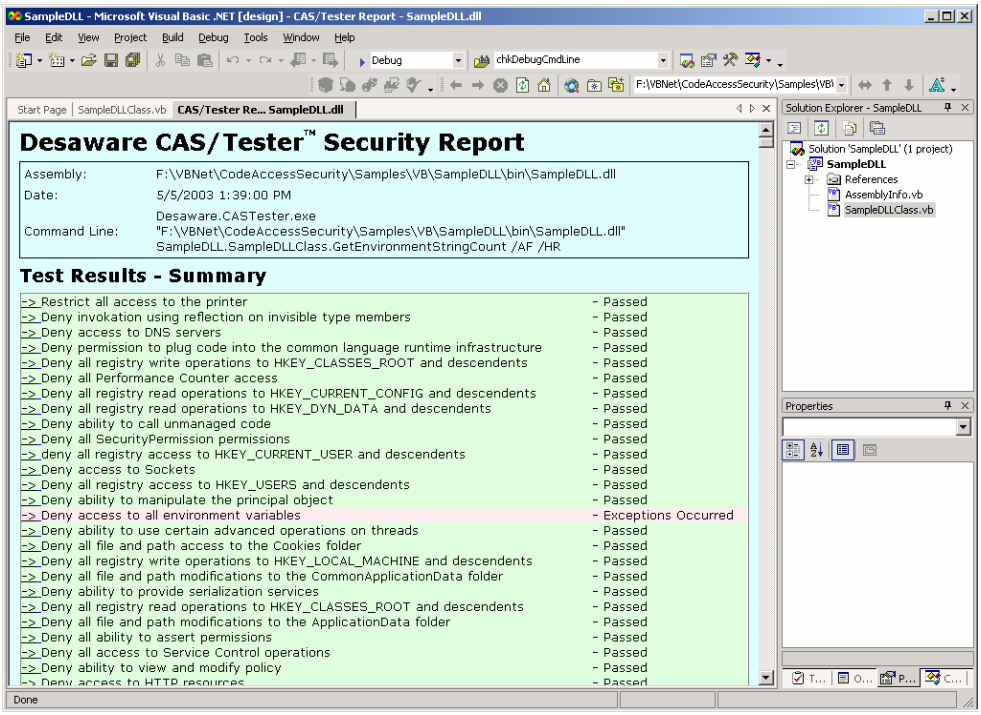

Figure 4
CAS/Tester Report

Header	This includes the name of the assembly being tested, the test date, and the command line options used for the test.
Script Errors	Any compilation errors in your test script will appear here.
Setup Errors	Any errors that occur while setting up a test will appear here. These usually reflect errors in command line parameters.
Test Results Summary	A summary of all tests and their results. Includes links to jump to a specific test result.
Test Results	Each test is listed in detail. The test result includes a detailed listing of exactly what permissions were applied, and which exceptions occurred during the testing.

A stack trace is included to help you identify where in your code the security error occurred. The stack trace should identify where in the test assembly or test script the exception took place. If your test assembly was compiled in Debug mode, the stack trace should also include the line number. We recommend testing assemblies that are compiled with debugging information.

Two report files are generated, an XML file containing the result data, and an HTML file that contains a report generated using the CasReport.xslt file. The XML report allows you to perform automated result analysis, including regression testing (to determine if different results occur after changes are made to the code).

Command Line Options

CAS/Tester supports a variety of command line options. In most cases an option includes both a full word and two letter syntax (thus /SingleTest is the same option as /ST). If an option includes a file name that has spaces, the file name should be enclosed in quotes.

/SingleTest=*test* /ST=*test*	Execute only one test. *test* specifies the name of the test (see appendix A).
/CommandLine=*command* /CL=*command*	When testing an executable that uses command line parameters, use this option to specify the command line to pass to the test executable. This option must be the last one on the CAS/Tester command line.
/TimeoutExect=*time* /TE=*time*	This is the time that CAS/Tester waits after running an executable before attempting to create the requested object. Default time is 1000 milliseconds.
/TimoutTest=*time* /TT=*time*	This is the maximum time for the test. After this time expires, CAS/Tester will attempt to close or shutdown the test program. CAS/Tester will attempt to close any open windows during the shutdown process. If it is unable to terminate the test, an error will be reported and testing will stop. Default time is 5000 milliseconds.
/AbortForm /AF	When specified, this option causes a form to be displayed that allows you to stop testing. Testing will stop at the end of the current test. This option cannot abort an ongoing test.
/AlternateReport=*filename* /AR=*filename*	By default, the CAS/Tester report is generated using the CAStester.xslt file provided by Desaware. Use this option to specify your own XSLT transform file.
/HideReport /HR	By default, the CAS/Tester report is displayed after the test by launching the default application associated with the .htm file extension. Use this option to NOT display the report after test completion.

Test Scripts

For basic testing, CAS/Tester provides a simple and effective way to test your assemblies. One common approach is to define a test function within the assembly that performs the tests that you want – in effect making the assembly self-testing.

However, if you wish to define custom tests, or to create an external test harness, you must use the scripting capability. Scripting requires a basic familiarity with writing software that uses Code Access Security.

To use scripting, instead of specifying an object and method on the command line, you specify a .vb or .cs source file.

The source file must contain three sections, command lines, the test setup, and the test script.

Command Lines

If the first line in the file is a comment, subsequent lines are checked for a '/' character. If found, the line is considered a command line parameter to the compiler. For example: to add additional references to the script, you would use the following lines at the start of the script file.

[VB .NET]

```
' CommandLines
' /r:system.serviceprocess.dll,System.Messaging.dll
```

[C#]

```
// CommandLines
// /r:system.serviceprocess.dll,System.Messaging.dll
```

References to your test assembly, and to the CAS/Tester program, are automatically included during script compilation.

Test Setup

The script file must contain a class named TestScript. The class must have three public HashTable fields as follows:

[VB .NET]

```
Public Class TestScript
    ' Contains objects derived from CodeAccessPermission
    Public DenyPermissions As New Hashtable()

    ' Contains objects derived from PermissionSet
    Public ApplyPermissionSets As New Hashtable()

    ' Contains optional descriptions for each test.
    Public TestDescriptions As New Hashtable()
```

[C#]

```
// Do NOT specify a namespace in your Test Script file

public class TestScript
{

    // Contains objects derived from CodeAccessPermission
    public Hashtable DenyPermissions = new Hashtable();

    // Contains objects derived from PermissionSet
    public Hashtable ApplyPermissionSets = new Hashtable();

    // Contains optional descriptions for each test
    public Hashtable TestDescriptions = new Hashtable();
```

- Every test must have a name, which is used as the key in the Hash Table.
- The DenyPermissions HashTable contains individual permissions that derive from CodeAccessPermission. During testing, each permission is denied in turn using the CodeAccessPermission.Deny method (unless overridden using the ICASUtility SetPermitOnly method).
- The ApplyPermissionSets HashTable contains permission sets that derive from PermissionSet. During testing, each permission set is denied in turn using the PermissionSet.Deny method (unless overridden using the ICASUtility SetPermitOnly method).
- The TestDescriptions HashTable should be loaded with descriptions for each test. This is the description that will appear in the test report.
- You should load the Hash Table objects during the class constructor.

[VB .NET]

```
    Public Sub New(ByVal HelperFunctions As ICASUtility)
```

[C#]

```
    public TestScript(ICASUtility HelperFunctions)
```

The constructor takes a ICASUtility object (provided by CAS/Tester). This object provides a number of helper methods.

ICASUtility members relating to permissions:

TestAssembly	This Readonly property returns the full path of the assembly being tested.
SetPermitOnly(*testname*)	When a test name is registered using this method (during the class constructor), instead of using the Deny method, the PermitOnly method is used to apply the Permission or PermissionSet.
GetLocalPermissionSets(*level*)	This method returns a HashTable containing copies of each PermissionSet defined at the specified level.
BuildPermitOnlyPermissionSet(*permission*)	This method takes the "Everything" permission set (which allows every operation), and substitutes the specified permission of the same type. Use this with SetPermitOnly, to restrict a permission.
GetLevels()	Returns a string array containing the names of the security levels defined on this system.

The following are examples of how to add permissions and permission sets to the constructor.

[VB .NET]

```
Public Sub New(ByVal HelperFunctions As ICASUtility)
    ' Deny unmanaged code permission
    DenyPermissions.Add("UnmanagedCode", New _
    Permissions.SecurityPermission( _
    Permissions.SecurityPermissionFlag.UnmanagedCode))
    TestDescriptions.Add("UnmanagedCode","Deny unmanaged code")


    ' Deny all access except for read access to one file
    ApplyPermissionSets.Add("DenyAllButOne", _
    HelperFunctions.BuildPermitOnlyPermissionSet(New _
    Permissions.FileIOPermission(FileIOPermissionAccess.Read Or _
    FileIOPermissionAccess.PathDiscovery, "c:\onefile.txt")))
    HelperFunctions.SetPermitOnly(" DenyAllButOne")
    TestDescriptions.Add("DenyAllButOne","Allow read onefile.txt")

End Sub
```

[C#]

```
public TestScript(ICASUtility HelperFunctions)
{
        // Deny unmanaged code permission
        DenyPermissions.Add("UnmanagedCode", new
Permissions.SecurityPermission(
Permissions.SecurityPermissionFlag.UnmanagedCode));
        TestDescriptions.Add("UnmanagedCode","Deny unmanaged code");

        // Deny all access except for read access to one file
        ApplyPermissionSets.Add("DenyAllButOne",
HelperFunctions.BuildPermitOnlyPermissionSet(new
Permissions.FileIOPermission(FileIOPermissionAccess.Read |
FileIOPermissionAccess.PathDiscovery, "c:\\onefile.txt")));
        HelperFunctions.SetPermitOnly("DenyAllButOne");
        TestDescriptions.Add("DenyAllButOne","Allow read onefile.txt");
}
```

The first example shows how to add a test that denies unmanaged code permission. Note how the TestDescriptions Hash Table is loaded with the description for the test.

The second example shows how to use a permit instead of deny. Let's say you want to block all file access except for the right to read file c:\onefile.txt. To deny all other files would be very hard – you'd have to create a list of all possible files other than c:\onefile.txt.

Instead, use the BuldPermitOnlyPermissionSet to create a permission set that allows everything, substituting a file permission that allows access only to c:\onefile.txt. When PermitOnly is applied to this permission set, you will allow full access in every permission type except for FileIOPermission, and in FileIOPermission will allow only access to c:\onefile.txt. Note the use of SetPermitOnly to set PermitOnly for this permission set.

Test Script

Your TestScript class should have a single method named Test, which is defined as follows:

[VB .NET]

```
Public Sub Test(ByVal TestAssembly As [Assembly], _
                ByVal HelperFunctions As ICASUtility)
        ' Do your testing here
End Sub
```

[C#]

```
public void Test(Assembly TestAssembly, ICASUtility HelperFunctions)
{
```

```
        // Do your testing here
}
```

The function takes two parameters: A reference to the test assembly, and a reference to the ICASUtility object described earlier. During this method, you should perform any tests you wish on the assembly.

By default, the requested security settings are applied before this method is called. However, it is likely that there will be times where you need your test script to run in full trust, and will wish to apply security yourself. To do so you must apply the DeferSecurity attribute to the Test method.

When you use DeferSecurity, you apply security using the following code:

[VB .NET]

```
    <DeferSecurity()> Public Sub Test(ByVal TestAssembly As [Assembly], ByVal
HelperFunctions As ICASUtility)
        If HelperFunctions.UsePermitOnly Then
            HelperFunctions.ApplyPermission.PermitOnly()
        Else
            HelperFunctions.ApplyPermission.Deny()
        End If
```

[C#]

```
    [DeferSecurity()] public void Test(Assembly TestAssembly, ICASUtility
HelperFunctions)
    {
        if (HelperFunctions.UsePermitOnly)
            HelperFunctions.ApplyPermission.PermitOnly();
        else
            HelperFunctions.ApplyPermission.Deny();
```

ICASUtility Members Relating to Test functions:

ApplyPermissions	This property returns the PermissionSet reflecting the current test.
UsePermitOnly	This property returns True if you should use PermitOnly instead of Deny on the current test.
SetTimerDelegate(*callback*)	Allows you to set a delegate to be called at frequent intervals during the test. A TimerCallbackDelegate is defined as follows: `Public Delegate Sub TimerCallbackDelegate(ByVal HelperFunctions As ICASUtility)`

	When testing a Windows forms based application (including UserControls), this delegate will be called on the same thread as the primary form or control. Thus it is safe during this delegate call to invoke methods on those objects. The callback delegate will not be called when message boxes are shown, or when a private message loop is running.
SetAsyncTimerDelegate(*callback*)	Identical to SetTimerDelegate, except that the callback function is called on a background thread. Calls will take place even when message boxes are shown or private message loops are running. Use this callback with caution and be sure to use appropriate synchronization techniques.
AddProgressNoteToReport(*note*)	This method adds a progress note to the output report for the current test.
TryToCloseWindow(*caption*)	This method finds the window with the specified caption and tries to close it.
SetWindowsFocus(*TargetForm*)	This method sets the Form specified as the topmost window on the system. This is needed if you are going to use SendKeys or other interface activity that depends upon the window being on top.
ClickControlOnForegroundWindow (*windowname*)	This method will send a mouse click to the control with the specified text on the topmost form. This is primarily useful for buttons.
ClickWindow (*windowname*)	This method will send a mouse click to the first window in the system it locates with the specified text.
ClickControl (*ctl*)	This method will send a mouse click to the specified Control. This method is mostly useful for buttons or similar controls.
ClickControl (*ctl, x, y*)	Similar to above with the addition that the mouse click occurs in the specified x and y coordinates of the Control.

Sample Test Scripts

CAS/Tester includes several test script samples demonstrating how to use test scripts to test class libraries, windows forms applications, and windows usercontrols. This section describes several of these test scripts and their uses. Sample test scripts are provided in both VB.NET and C#.

SampleDLL is a sample class library that exposes several functions that require permission to access certain resources. You can test this file directly with CAS/Tester or by using a Test Script. Note that when testing the file directly, CAS/Tester cannot call functions that require parameters. Please review the functions exposed by the SampleDLL project. This will help identify which functions may encounter exceptions when certain permissions are denied.

The **TestScript sample** project demonstrates how to create a test script file that can be compiled by CAS/Tester to test an assembly. The Class1 file contains the TestScript class which exposes a Public Sub named Test. This is the function where you would place your test script code.

The remainder of the TestScript project demonstrates how you can test the same functions as your test script. The TestScript project references the SampleDLL.dll and Desaware.CASTester.dll assemblies. The Desaware.CASTester.dll file exposes some functions that the test script uses. Referencing this file in your TestScript project allows you to catch compile time errors while editing your script file, rather than having CAS/Tester catch

them during testing. We recommend starting with this example and calling some of the exposed functions from this assembly using the Single Test method.

The **InterestApp_Before and InterestApp_After** folders contain two versions of the same simple mortgage calculation application. One of the controls on the form is a button that uses HTTP to retrieve a text file containing the "latest interest rate" value. This will cause an exception in cases where the user does not have security permission to access HTTP resources. The script files are written to test situations where access to network resources are limited (by pressing this button).

In the InterestApp_Before application there is no error checking, and when HTTP resources are not available an exception occurs. In the InterestApp_After application, appropriate error checking was added informing the user when the button can not be used and therefore, no exceptions take place. The InterestCalculatorScript.vb and InterestCalculatorScriptA.vb script files found in each folder contain the test script code to do the testing for these applications.

The **TextViewer1_Before and TextViewer1_After** folders contain two versions of the same sample application. The program allows you to open, modify, and save a text file. It is designed to demonstrate many of the common program features that might be limited by changes to the user's code access security, such as file access, registry access, and internet access. It includes many common interface elements, such as menus, dialog boxes, buttons and edit boxes to show how to create test scripts that interact with them. It also includes methods to launch background threads and other applications, illustrating how CAS/Tester behaves in such situations.

The TextViewer1_Before application has no error checking. The TextViewer1_After application includes error trapping, with script modifications to handle such errors. The TextViewerScript_Before.vb and TextViewerScript_After.vb script files found in each folder contain the test script code to do the testing for these applications.

The **ControlSample_Before** and **ControlSample_After** folders contain two versions of the same user control and demonstrates how to test Windows User Controls using CAS/Tester test scripts. The ControlSample user control is a simple control consisting of a single button which, when pressed, will call an unmanaged API function to change the text of the parent window on which the control is placed.

When CAS/Tester detects that it is testing a Windows forms control, it creates a blank Windows Form on which the control is placed. If you are running an automatic test, the control is placed on the form. If you are creating a test script (as is the case with the sample project) then that task is left to you. The test script will load and place the control, set required properties to correctly show and configure the control, and then activate the control. The ControlScript_Before.vb and ControlScript_After.vb script files found in each folder contain the test script code to do the testing for these applications.

Additional Notes on Test Scripts

The preferred method of testing Windows forms applications or controls is to create a Public Function which can then be called from the CAS/Tester test script. The test script is limited in the manner in which it can interact with such solutions, and the test script may itself be limited by security tests meant for the solution. For example, most of the ways for interacting with Windows Forms (such as the .NET SendKeys Framework object or the Click methods provided by the CAS/Tester helper functions library) involve using unmanaged calls - if you are testing a solution under cases where unmanaged calls are not allowed then the test script will fail.

When using test scripts to test Windows user controls, components or class libraries, you must manually create the objects you wish to test in your test scripts. For Windows user controls, you must also make them visible.

For Windows forms applications, remember that CAS/Tester executes the application before calling your test script. This means that any forms normally brought up by your application will already be loaded and probably visible before your test script begins. Use the ActiveForm property to obtain a reference to the active form for your application.

Additional Samples

The **SampleConsoleApp** folder contains a sample Windows console application. It takes as input a command line argument value from 0 to 4 and calls functions requiring a different permission set depending on the input value. Please refer to the sample project for a list of functions that are called. You can test this sample by specifying a Command Line Argument value from 0 to 4 when testing this application.

File Descriptions

The following table describes the files included with CAS/Tester. Files compiled with Visual Studio version 1.1 have a "11" appended to their file names but are otherwise identical in nature to their 1.0 compiled files.

Desaware.CASTester.exe	Primary CASTester application. This application can be launched using the command line, the CASTester Launcher Add-In, or the CASTester Launcher standalone application.
Desaware.MachineLicense.dll	Desaware Licensing System assembly file. Required in order to run CASTester.
CASTester10.dlsc	Desaware Licensing System license file for your specific system. This file is created by the CASTester installation. Required in order to run CASTester.
CASTesterInstaller.exe	Used when installing and uninstalling CASTester.
CASTesterLauncher.exe	CASTester Launcher standalone application. Used to configure options and to launch CASTester.
CASTesterAddIn.dll	Visual Studio Add-In version of the CASTester Launcher application. Used to configure options and to launch CASTester.
CASTesterAddIn.tlb	CASTester Launcher Add-In Type Library. Used by the CASTester Launcher Add-In.
CASTesterUI.dll	CASTester Launcher User Interface assembly. Used by CASTester Launcher.
CASTester.pdf	Primary CASTester manual in Adobe Acrobat file format.
Readme.rtf	CASTester Readme file. Please read this file for latest information.
Desaware.CASTester.dll	This assembly contains the ICASTUtility interface definition. You can reference this file in your Test Script projects to enable Visual Studio Intellisense for your Test Script file.
TestScriptTemplate.*	Visual Basic .NET and C# template files for creating Test Scripts. These files are installed to the Samples folder.
SampleDLL	Sample .NET Class Library Assembly. This file is used by the Test Script samples to demonstrate writing a Test Script to test Class Libraries.
TestScript	Sample Test Scripts demonstrating how to create objects from class libraries and call their functions.
SampleConsoleApp	Sample Console Application for use in testing .NET Console Applications with CASTester.
InterestApp_Before	Sample Windows Forms Application and Test Script. Demonstrates how to create a Test Script to test a Windows Forms Application.

InterestApp_After	Similar to the InterestApp_Before sample. This sample includes exception handling to handle exceptions for situations where certain permissions are not permitted for the application.
TextViewer1_Before	Sample Windows Forms Application and Test Script. Demonstrates how to create a Test Script to test a Windows Forms Application.
TextViewer1_After	Similar to the TextViewer1_Before sample. This sample includes exception handling to handle exceptions for situations where certain permissions are not permitted for the application.
ControlScript_Before	Sample User Control and Test Script. Demonstrates how to create a Test Script to test a User Control.
ControlScript_After	Similar to the ControlScript_Before sample. This sample includes exception handling to handle exceptions for situations where certain permissions are not permitted for the user control.

Licensing

Like any piece of software, CAS/Tester is subject to a very official looking and legalistic license agreement. Here, in plain language, is how it works in a nutshell:

- CAS/Tester is licensed per machine. Every machine you run it on requires a separate license and install key (contact us about quantity and site license discounts).
- If your machine hosts multiple operating systems, you only need a single license for all of the operating systems on that machine. We use a deferred activation licensing scheme (see below), and if we see that an install key has been used on multiple machines the program will initially warn you, and then later disable, use of that installation key.
- CAS/Tester is not designed to run over a network share.
- You may not run CAS/Tester as a service for use by multiple clients. (You wouldn't want to anyway, because doing so represents a severe security risk to your server – CAS/Tester itself must run in full-trust and provides no protection from the programs that are being tested.)
- CAS/Tester uses a deferred activation licensing scheme. During or after installation, CAS/Tester will contact Desaware's licensing server and register use of the installation code for your system. The information sent to the server uses cryptographic hashing to allow us to detect use of the install code on multiple systems without use of any information that could identify a system – thus protecting your privacy. You can read more about our licensing at www.desaware.com /DlsL2.htm.

More Facts

- The following are additional facts that will help you with using CAS/Tester.
- When using the /CL option to specify a command line to an application, the first parameter will always be the path to Desaware.CASTester.exe, not the path to the assembly being tested.
- The most common reason for CAS/Tester being unable to terminate a test is if your code does not return from the test function in the time specified by the /TimeoutTest option.
- Assemblies you test should be installed under full trust during testing.
- CAS/Tester must run in full trust.
- The Test function you specify can be private if it returns no parameters (void or Sub). If it returns a value, the function must be public (even if it is in a .exe file).
- You may choose to write your test script in either Visual Basic .NET or C# to test any .NET assembly. You are not required to write your test script in the same language as the compiled assembly being tested.

Appendix A – List of Default Permission Tests

DefaultPrinting	Allow general printing to the default printer
EventLogBrowseOnly	Allow only Browsing (read only) on Event Logs
EventLogInstrumentOnly	Allow only Instrumentation (read & write) on Event Logs
NoAppDataFileModify	Deny all file and path modifications to the ApplicationData folder
NoCommonAppDataFileModify	Deny all file and path modifications to the CommonApplicationData folder
NoCookiesFileAccess	Deny all file and path access to the Cookies folder
NoCookiesFileModify	Deny all file and path modifications to the Cookies folder
NoDirectoryServiceAccess	Deny all access to Directory Services
NoDNSAccess	Deny access to DNS servers
NoEnvAccess	Deny access to all environment variables
NoEventLogAccess	Deny Event Log access
NoFileAccess	Deny all access to files (accept reading the test assembly)
NoFileDlgAccess	Deny the ability to access files via file dialogs (OpenFile method)
NoFileDlgSave	Deny the ability to save files through file dialogs (OpenFile method)
NoFileModify	Deny all file and path modifications
NoIsolatedStorageAccess	Deny access to a private virtual file system
NoLocalAppDataFileModify	Deny all file and path modifications to the LocalApplicationData folder
NoLocalFileModify	Deny all file and path modifications on the Local Computer

NoMSQAccess	Deny access to the Message Queue
NoOleDBAccess	Deny all access to an OLE DB data source including blank passwords
NoOleDBAccessNoBlankPassword	Deny all access to an OLE DB data source, not including blank passwords
NoPathDiscovery	Deny all path discovery for all files and directories
NoPerfCounterAccess	Deny all Performance Counter access
NoPersonalFileAccess	Deny all file and path access to the Personal folder
NoPersonalFileModify	Deny all file and path modifications to the Personal folder
NoPrintAccess	Deny all access to the printer
NoReflectionEmitAccess	Deny reflection for emitting metadata and intermediate language (MSIL)
NoReflectionMemberAccess	Deny invocation using reflection on invisible type members
NoReflectionTypeAccess	Deny reflection for invisible type information
NoRegAccess	Deny all registry access
NoRegAccessOnHKEY_CLASSES_ROOT	Deny all registry access to HKEY_CLASSES_ROOT and descendents
NoRegAccessOnHKEY_CURRENT_CONFIG	Deny all registry access to HKEY_CURRENT_CONFIG and descendents
NoRegAccessOnHKEY_CURRENT_USER	deny all registry access to HKEY_CURRENT_USER and descendents
NoRegAccessOnHKEY_DYN_DATA	Deny all registry access to HKEY_DYN_DATA and descendents
NoRegAccessOnHKEY_LOCAL_MACHINE	Deny all registry access to HKEY_LOCAL_MACHINE and descendents
NoRegAccessOnHKEY_PERFORMANCE_DATA	Deny all registry access to HKEY_PERFORMANCE_DATA and descendents
NoRegAccessOnHKEY_USERS	Deny all registry access to HKEY_USERS and descendents

NoRegModifyOnHKEY_CLASSES_ROOT	Deny all registry modifications to HKEY_CLASSES_ROOT and descendents
NoRegModifyOnHKEY_CURRENT_CONFIG	Deny all registry modifications to HKEY_CURRENT_CONFIG and descendents
NoRegModifyOnHKEY_CURRENT_USER	Deny all registry modifications to HKEY_CURRENT_USER and descendents
NoRegModifyOnHKEY_LOCAL_MACHINE	Deny all registry modifications to HKEY_LOCAL_MACHINE and descendents
NoRegModifyOnHKEY_USERS	Deny all registry modifications to HKEY_USERS and descendents
NoRegReadOnHKEY_CLASSES_ROOT	Deny all registry read operations to HKEY_CLASSES_ROOT and descendents
NoRegReadOnHKEY_CURRENT_CONFIG	Deny all registry read operations to HKEY_CURRENT_CONFIG and descendents
NoRegReadOnHKEY_CURRENT_USER	Deny all registry read operations to HKEY_CURRENT_USER and descendents
NoRegReadOnHKEY_DYN_DATA	Deny all registry read operations to HKEY_DYN_DATA and descendents
NoRegReadOnHKEY_LOCAL_MACHINE	Deny all registry read operations to HKEY_LOCAL_MACHINE and descendents
NoRegReadOnHKEY_PERFORMANCE_DATA	Deny all registry read operations to HKEY_PERFORMANCE_DATA and descendents
NoRegReadOnHKEY_USERS	Deny all registry read operations to HKEY_USERS and descendents
NoRegWriteOnHKEY_CLASSES_ROOT	Deny all registry write operations to HKEY_CLASSES_ROOT and descendents
NoRegWriteOnHKEY_CURRENT_CONFIG	Deny all registry write operations to HKEY_CURRENT_CONFIG and descendents
NoRegWriteOnHKEY_CURRENT_USER	Deny all registry write operations to HKEY_CURRENT_USER and descendents
NoRegWriteOnHKEY_LOCAL_MACHINE	Deny all registry write operations to HKEY_LOCAL_MACHINE and descendents
NoRegWriteOnHKEY_USERS	Deny all registry write operations to HKEY_USERS and descendents
NoSecurityAccess	Deny all SecurityPermission permissions
NoSecurityAssertion	Deny all ability to assert permissions

NoSecurityCtrlAppDomain	Deny ability to create and manipulate an AppDomain
NoSecurityCtrlDomainPolicy	Deny ability to specify domain policy
NoSecurityCtrlEvidence	Deny ability to provide or alter evidence
NoSecurityCtrlPolicy	Deny ability to view and modify policy
NoSecurityCtrlPrincipal	Deny ability to manipulate the principal object
NoSecurityCtrlThread	Deny ability to use certain advanced operations on threads
NoSecurityInfrastructure	Deny permission to plug code into the common language runtime infrastructure
NoSecurityRemotingConfig	Deny permission to configure Remoting types and channels
NoSecuritySerializationFormat	Deny ability to provide serialization services
NoSecurityUnmanagedCode	Deny ability to call unmanaged code
NoServiceControlAccess	Deny all access to Service Control operations
NoSocketAccess	Deny access to Sockets
NoSQLAccess	Deny all access to a data source, including blank passwords
NoSQLAccessNoBlankPassword	Deny all access to a data source, not including blank passwords
NoSystemFileModify	Deny all file and path modifications to the System folder
Nothing	Nothing: Denies all resources. Execution not blocked.
NoUIAccess	Deny UI access to windows and the clipboard
NoWebAccess	Deny access to HTTP resources
SafePrinting	Allow safe printing to the default printer
UIClipBoard	Deny UI access to Windows.

UIOwnClipBoard	Allow access only to your own clipboard - No Windows access
UISubWindows	Allow UI access to safe sub Windows only - Own clipboard access
UITopLevelWindows	Allow UI access to top level Windows only - Own clipboard access
UIWindows	Deny UI access to clipboard

Additional Default Permission Tests for Visual Studio 1.1

NoASPNetHostingAccess	Deny all access to ASP.NET hosted environments
ASPNetHostingHighLevel	Allow high level of access to ASP.NET hosted environments
ASPNetHostingMediumLevel	Allow medium level of access to ASP.NET hosted environments
ASPNetHostingLowLevel	Allow low level of access to ASP.NET hosted environments
ASPNetHostingMinimalLevel	Allow medium level of access to ASP.NET hosted environments
ASPNetHostingNoLevel	Allow no level of access to ASP.NET hosted environments

Other Sources of Information

www.desaware.com

Desaware's web site includes numerous technical articles on all aspects of Windows development. Be sure to also peruse the FAQ and support section for this product.

Moving to VB.Net: Strategies, Concepts and Code, 2nd Edition

Written by Daniel Appleman (president of Desaware) and published by Apress (ISBN# 1 1-59059-102-X).

VB.Net is not Visual Basic. COM+2.0 is not COM.

Porting is stupid. These are just a few of the things you'll learn as Dan takes you on a journey unlike any other into the world of VB.Net. He tackles strategic issues to help you determine when and whether to deploy VB.Net.

As always, Dan teaches the core concepts such as inheritance and multithreading, where VB6 programming habits can lead to costly design and development errors. And he covers the language changes to help you adapt to, and understand, this new environment. Updated to include sample code for VS 2002 and 2003. Moving to VB.Net

Dan Appleman's Win32 API Puzzle Book and Tutorial for Visual Basic Programmers

Written by Daniel Appleman (president of Desaware) and published by Apress (ISBN# 1-893115-01-1). Appleman's Win32 API Guide covers 700 API functions. This book covers the other 7800. How? By teaching you everything you need to know to read and understand the Microsoft C documentation and create correct API declarations for use in Visual Basic. Presented in an entertaining puzzle/solution format that challenges you to solve real world API problems. In depth tutorials take you behind the scenes to understand what really happens when you call an API function from VB.

Dan Appleman's VB Programmer's Guide To The Win32 API

Written by Daniel Appleman (president of Desaware) and published by McMillan, (ISBN 0-672-31590-4) - this sequel to the original 16 bit API Guide applies the same philosophy to teaching the Win32 API to developers using Visual Basic and VBA based applications. With more examples, more functions, more tutorial style explanations and a full text searchable electronic 6.

Available at most good bookstores, or directly from Desaware at a 20% discount - call (408) 377-4770. Visual Basic Programmer's Guide to the Win32 API

An upgrade CD is available for owners of the "PC Magazine's Visual Basic Programmer's Guide to the Win32 API" ISBN: 1-56276-287-7 for $24.99 + s&h directly from Desaware. Refer to our web site at www.desaware.com for additional information.

Dan Appleman's Developing COM/ActiveX Components with VB 6.0

Written by Daniel Appleman (president of Desaware) and published by McMillan, (ISBN 1-56276-576-0) - this book is designed for those programmers interested in using Visual Basic's object oriented technology to develop ActiveX components including EXE and DLL servers, ActiveX controls and ActiveX documents. Unlike many books that simply rehash the Visual Basic documentation, this one serves as a commentary to clarify and extend the documentation. Of special interest to VersionStamper customers will be the chapters on OLE and COM technology that will help them further understand the process of registering components, and the chapters on versioning and licensing.

The VB6 version also includes two new chapters on IIS Application development.

Available at most good bookstores, or directly from Desaware at a 20% discount - call (408) 377-4770. Developing COM/ActiveX Components with VB6

Dan Appleman's Ebook Series

Hijacking .NET: Using undocumented .NET internals
VB.Net or C#: Which to Choose?
Regular Expressions with .NET
Exploring .NET - A collection of articles on .NET topics.
Tracing and Logging in .NET
Obfuscating .NET: Protecting your code from prying eyes.
Introduction to NT Security (VB6)

Windows API Online Help

The Professional Edition of Visual Basic includes Win31api.hlp and/or win32api.hlp - an online help reference for all API functions. These functions are declared in C and do not consider Visual Basic compatibility issues, however the information in chapter 3 of the Visual Basic Programmer's Guide to the Windows API (chapters 3 and 4 of the 32 bit book) will provide you with information on how to translate these functions to Visual Basic.

Microsoft's Developers Network CD Rom

This amazing CD-ROM and web site (http://msdn.microsoft.com) contain a wealth of information and sample code, plus the latest Visual Basic knowledge base.

Microsoft's Windows Software Development Kit and Win32 Software Development Kit

The sample code is all in C, but by the time you've read the Visual Basic Programmer's Guide to the Windows API or Win32 API, you'll know enough to be able to translate the C code to Visual Basic.

Desaware Product Descriptions

Thank you for your purchase of this Desaware product. We have additional quality software to enhance your programming efforts. Please visit our web site at www.desaware.com for detailed descriptions and product demos.

SPYWORKS Professional 7.0

SpyWorks in a nutshell? Impossible!

You're going to want to download the SpyWorks demo to even begin to understand its capabilities. This product has been evolving for several years, and it includes so many features it's hard to know where to begin. SpyWorks is a VB power tool. When you need to override VB's default behavior or to extend VB's functionality, you will want to use SpyWorks.

Do *That* in Visual Basic??

Want to put VB to the test? Want to learn advanced programming techniques? Want to keep the productivity of VB and have the functionality of C++? SpyWorks contains the low level tools that you need to take full advantage of Windows. Here are just a few of the features of this multi-faceted software package. For instance, have you ever wanted to detect keystrokes on a system-wide basis or detect when an event occurs in another application or thread using subclassing or hooks? SpyWorks can help you solve these problems by letting you tap into the full power of the Windows API without having to be an expert. SpyWorks lets you export functions from VB DLL's so that you can create function libraries, control panel applets, and NT Services. With its ActiveX extension technology, you can call and implement interfaces that VB5 or 6 do not support. SpyWorks includes the Desaware API Class Library, which assists programmers in taking advantage of the hundreds of functions that are built into the Windows API. SpyWorks is available in either the Professional (Pro) or Standard edition.

The Professional Edition includes .NET support for keyboard hooks, window hooks and subclassing (including cross-task subclassing) with examples in both Visual Basic.NET and C#. Additionally, a WinSock component with comprehensive VB source code that gives you complete control for Internet/intranet programming.

Other features are the NT Service Toolkit *Light Edition*. This application is a subset of the Desaware NT Service Toolkit product. It allows a developer to create true NT services using Visual Basic. A background thread component that allows you to easily create objects that run in a separate background thread.

It also contains extensive sample code.

- The Professional Edition includes the Winsock Library, NT Service support and many other additional features & samples. SpyWorks 2.1 (VBX Edition) is included in the Pro Edition.
- Supports VB 4, 5 & 6, Windows 95, 98, 2000, NT and ME depending upon which version (or edition) of SpyWorks.

STATECODER 1.0

A .NET class framework that makes it easy to create and support powerful state machines using VB .NET or C#. Dramatically improves the reliability of applications, components and services that make use of the multithreading and asynchronous features of .NET.

VERSIONSTAMPER 6.5

Distributing Component-Based Applications? Beware DLL HELL!

You've distributed your application and it's working fine. But your end user is still in charge of their system. What happens when they install a program that overwrites a component that your software needs to run? Can you verify that your users have the correct files required by your application? Can you really afford to spend two hours on the phone trying to figure out exactly what went wrong? Now you can easily avoid component incompatibilities by adding VersionStamper to your toolkit. It lets you check the versions of your program's components on your end user's system, and correct the problem.

You are in control!

DLL Hell is a big problem, and with VersionStamper you can be in control of how this problem is detected and corrected. You determine dependency scanning (file size, date, version or other parameter), how and when the dependency scanning is done (upon start up, at midnight, at user's discretion), and how you want the problem resolved (automatically, an email message to your help desk, from a dependency list on your web site and more). This means you can handle versioning problems as simply as using a message box to call tech support, or even automatically updating the invalid components over the internet or corporate network. Imagine your application updating itself without user (or programmer) intervention! Imagine the hours and money saved in tech support calls! You can even use VersionStamper for incremental updates and bug fixes.

Is This For Real?

No, you don't have to pay a fortune in distribution fees - there are no run-time licensing fees. VersionStamper comes with a great deal of sample code. Don't distribute a component-based application without it!

- Checks the versions of your dependent files and notifies you or the user of potential problems.
- Internet extensions allow you to update versions across the Internet/intranets.
- Cool and USEFUL sample programs show you how it works.

Includes VB source code for the VersionStamper components that you can use in your applications.

NT SERVICE TOOLKIT 2.0 COM Edition, .NET Edition

Create a fully featured service in minutes using Visual Basic – even debug your service using the Visual Basic environment! Supports all NT service options and controls. Adheres to all Visual Basic threading rules. Background thread support allows easy waiting on system and synchronization objects. Client requests supported on independent threads for excellent scalability, with client impersonation available allowing services to act on behalf of clients in their own security context. Client requests and service control possible via COM/COM+/DCOM.

Simulation mode for testing as an independent executable. Create control panel applets for service control and other purposes.

DESAWARE EVENT LOG TOOLKIT 1.0

Visual Basic allows you to log events to the NT/2000 event log, but does not allow you to create custom event sources - so every event belongs to the application VB runtime, descriptions are limited, and event categories unavailable. Even if you use the API to log events, creating custom event sources for your application is not supported by VB, and is difficult with C++.

Desaware's new Event Log Toolkit makes creation of event sources easy, and provides all the tools needed to create and log custom events. Now your applications and services can support event logs in a professional manner, as recommended by Microsoft

STORAGETOOLS ver 3.0

StorageTools is your key to the OLE 2.0 Structured Storage Technology. Structured Storage allows you to create files that organize complex data easily in a hierarchical system. It is like having an entire file system in each file. OLE 2.0 takes care of allocating and freeing space within a file, so just as you need not concern yourself with the physical placement of files on disk, you can also disregard the actual location of data in the file. Additionally, with its support for transactioning you can easily implement undo operations and incremental saves in your application. StorageTools allows you to take advantage of the same file storage system used by Microsoft's own applications. In fact, we include programs (with Visual Basic source code) that let you examine the structure of any OLE 2.0 based file so that you can see exactly how they do it!

StorageTools includes registration database controls for Windows NT, Windows 2000/XP, Windows 95 & 98. Plus, a simple resource compiler (with source) so that you can create your own .RES files for use with Visual Basic and more. 16 & 32 bit COM/ActiveX and .NET.

New for version 3.0! StorageTools 3.0 includes .NET support for accessing OLE Structure Storage from .NET assemblies.

DESAWARE ACTIVEX GALLIMAUFRY Ver. 2

What is it?

gal·li·mau·fry (gàl´e-mô¹frê) noun
plural gal·li·mau·fries
A jumble; a hodgepodge.

[French galimafrée, from Old French galimafree, sauce, ragout : probably galer, to make merry. See GALLANT + mafrer, to gorge oneself (from Middle Dutch moffelen, to open one's mouth wide, of imitative origin).]
(From The American Heritage® Dictionary of the English Language, Third Edition copyright © 1992 by Houghton Mifflin Company)

What does a Twain control, spiral art program, set of linked list classes, a quick sort routine, a hex editor and a myriad of other custom controls have in
common?

They are all part of Desaware's ActiveX Gallimaufry.

You'll find most of these controls useful, the rest entertaining – but we guarantee that you'll find them all educational, because they come with complete Visual Basic 6.0 source code.

Curious?

Want to learn some advanced API programming techniques? Visit our web site for a full description and demo.

THE CUSTOM CONTROL FACTORY V 4.0
The Custom Control Factory is a powerful tool for creating your own animated buttons, multiple state buttons, toolbars and enhanced button style controls in Visual Basic and other OLE control clients, without programming. With 256 & 24 bit color support, automatic 3D backgrounds, image compression, over 50 sample controls and more. Plus MList2 - an enhanced listbox control. 16 & 32 bit ActiveX controls and 16 bit VBXs included.