

Volume

1

UNIVERSITY OF WEST FLORIDA

Element Suite

User and Development Manual

ELEMENT SUITE

User and Development Manual

University of West Florida
COT6931 2647
Dr. Donna Lohr
Fall 2012

Table of Contents

What It Is	1
What It Does	1
How To Install	1
Using Element Suite	2
Earth	2
Air	2
Fire	2
Water	2
Resources	2
Troubleshooting/FAQs	3
Developing Addins	4
Available Shared Services	4
Pre-requisites and Tools	5
Creating Addin Assemblies	5
Addin Assembly	5
Distributed Assembly	6
Distributing The Work	7
Index	8

What It Is

Element Suite is a project to create an extensible security analysis application that provides support for distributed computing.


Element Suite is a project to create an extensible security analysis application that provides support for distributed computing. Many security and utility tools exist which provide a variety of security tools; however, each tool has been implemented in a variety of ways and thus do not provide a consistent user experience. Additionally, the tools either lack distributed computing capabilities or do not provide them in a consistent fashion.

What It Does

The Element Suite project solves the user experience and distributed computing consistency problem with two distinct applications. The main client application, Element Suite provides a shell application with a set of application programming interfaces (APIs) that developer may use to create a variety of security and utility oriented tools. The second application called Element Suite Moon may be installed on computers with available downtime such as in a computer lab to provide additional computing power for addins that are computationally intensive.

How To Install

- Pre-requisites
 - Microsoft .NET Framework 4.5 Full Install
<http://www.microsoft.com/en-us/download/details.aspx?id=30653>

ICON KEY Insert Code

- First, go to <https://github.com/bartsipes/ElementSuite> to retrieve the latest installers for Element Suite and Element Suite Moon. The website will link to the latest source code with the appropriate installers.
- The installer files will be bundles as .iso or .zip files titled “Element Suite Installer” and “Element Suite Moon Installer.” Download whichever format is easier to use.
- Install Element Suite using “Element Suite.msi” on the primary computer where you desire to run the application.
 - To install addins to Element Suite, go to the installation directory (%PROGRAMFILES(X86)%\5element\Element Suite\) and open the Addins folder.
 - Place the *.dll files for the addin into the Addins folder. The next time Element Suite starts the addin(s) will be automatically loaded into the application.
- Install Element Suite Moon using “Element Suite.msi” on the primary computer as well as any other computers with available computational resources.

Using Element Suite

All tools for Element Suite are setup as addins to the application. Each addin is broken up into one of four categories or “elements.”

- Earth – the utility group. Addins under this element should provide practical security or general purpose utilities.
- Air – wireless and networking. Addins under this element should provide utilities that interact with networks. This includes both wired and wireless networking.
- Fire – offense. Addins under this element should provide capabilities that actively seek out problems. These addins might be penetration testing tools or password decryption utilities.
- Water – defense. Addins under this element should provide tools which protect a user from malware, security flaws, or other intrusions.

Resources


For the latest version of Element Suite or to view support information please visit the project website at <http://bartsipes.github.com/ElementSuite/>.

Any issues or feature requests may be submitted using the issues section under the GitHub project website. <https://github.com/bartsipes/ElementSuite/issues>

Troubleshooting/FAQs

- Do I need to configure my network in any way?
 - Element Suite by default communicates over port 51360 for UDP traffic to initialize the Moon clients and then using TCP over ports 51361 and 51362 to facilitate performing distributed work. Just ensure that your firewall has been configured to permit traffic over these ports and that all computers running Element Suite and Element Suite Moon are on the same subnet.
- How can I change the default communication ports?
 - For Element Suite, the distributed work facilitation ports can be easily changed.
 - Open Windows Explorer and go to the Element Suite installation folder. By default this is %PROGRAMFILES(X86)%\5element\Element Suite\.
 - Edit the ElementSuite.UI.exe.config file.
 - Set the “ElementSuite.DistributedWorkQueuePort” and “ElementSuite.DistributedWorkQueueContextPort” appSettings to the needed ports. The values may be any port available on your network, typically between 49152 and 65535, provided the two ports are different from each other.

```
<configuration>
  <appSettings>
    <add key="ElementSuite.DistributedWorkQueuePort"
value="51362"/>
    <add key="ElementSuite.DistributedWorkQueueContextPort"
value="51361"/>
  </appSettings>
</configuration>
```

 FIGURE 1.1 setting the distributed work queue port.

- These ports are defined only in Element Suite and are dynamically passed to the Element Suite Moon clients.
- For the UDP broadcast port, the configuration for Element Suite and Element Suite Moon will need to be changed.
 - Open Windows Explorer and go to the Element Suite installation folder. By default this is %PROGRAMFILES(X86)%\5element\Element Suite\.
 - Edit the ElementSuite.UI.exe.config file.
 - Set the port on the endpoint address to the desired port. The port may be any port available on your network, typically between 49152 and 65535, provided it is different from the distributed work queue ports.

```
<configuration>
  <system.serviceModel>
    <client>
      <endpoint address="soap.udp://239.255.255.255:51360/" ... >
    </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

 FIGURE 1.2 setting the client endpoint address.

- Repeat the above process on all installations of Element Suite Moon by editing the ElementSuite.Moon.UI.exe.config file.
- The default location is %PROGRAMFILES(X86)%\5element\Element Suite Moon\.

Developing Addins

The process for creating addins for Element Suite involves interfaces and the managed extensibility framework (MEF), both components of the .Net Framework. All needed and available interfaces for addin development are documented within the GitHub project pages at <http://bartsipes.github.com/ElementSuite/API%20Documentation/index.htm>. The actual addin which displays an interface in Element Suite will implement [ElementSuite.Addin.Interface.IAddin](#). Addins which utilize the distributed APIs will have a separate Visual Studio project with code that implements the [ElementSuite.Common.Interface.IWorkCommand](#) interface. Both assemblies that are created need to have a strong name. See <http://msdn.microsoft.com/en-us/library/xc31ft41.aspx> for details on how to strongly type assemblies.

In the [ElementSuite.Common.Interface](#) namespace, Element Suite provides five different shared services accessible via the [IServiceLocator](#) interface which is passed to the class implementing [IAddin](#) when the [Initialize](#) method is called. All initialization code should be placed in the Initialize method and this method will be called only once when the addin is loaded. Within the Initialize method, the Resolve method on the [IServiceLocator](#) object may be called to retrieve and utilize any of the available services.

Available Shared Services

- [IFileService](#) – Create and access files and directory that are located in isolated storage specific to the assembly calling it.
- [ILoggingService](#) – Provides an interface for logging errors, warnings, and debug information. The service utilizes [log4net](#) which can be configured in the

ElementSuite.UI.exe.config file. See the log4net config samples <http://logging.apache.org/log4net/release/config-examples.html> for details on how to modify the Element Suite configuration file to meet your needs.

- [IMessageService](#) – Provides a simple set of APIs to interact with the end user via modal windows.
- [IResourceService](#) – Creates an [IResourceStore](#) that provides preservation of basic objects in a key value pair fashion.

Pre-requisites and Tools

- Visual Studio Express 2012 for Windows Desktop
<http://www.microsoft.com/visualstudio/eng/downloads#d-2012-express>
- Or Visual Studio 2012 Professional or greater
- Microsoft .NET Framework 4.5 Full install
<http://www.microsoft.com/en-us/download/details.aspx?id=30653>

Creating Addin Assemblies

Addin Assembly

- First, install Element Suite
- Open Visual Studio and create a new Class Library.
- Add a project reference to System.ComponentModel.Composition .NET Framework assembly
- Add projected references to ElementSuite.Addin.dll and ElementSuite.Common.dll
 - These dlls are located in the default Element Suite installation directory (%PROGRAMFILES(X86)%\5element\Element Suite\ by default).
- Create a class which implements [ElementSuite.Addin.Interface.IAddin](#)
- Add the following attribute to the class. This attribute exposes your class via the managed extensibility framework to Element Suite.

```
[System.ComponentModel.Composition.Export(typeof(ElementSuite.Addin.Interface.IAddin))]
```

Add initialization code to create the user interface for your addin by setting the Launch property of the IAddin interface. For example (launch is the backing field for the Launch property):


```
public void Initialize(IServiceLocator serviceFactory)
{
    launch = new MenuExtension() {
        Command = new RelayCommand((obj) => {
```



```

        var workBench = Factory.Resolve<IWorkbenchService>();
        var tabItem = new WorkbenchTab("Test Addin Title");
        tabItem.Content = new TestUserControl(serviceFactory);
        workBench.Add(tabItem);
    }},
    Name = "Test Addin"
};
}

```

 FIGURE 1.3 setting the launch property for addin.

- The [MenuExtension.Command](#) property is of type [ICommand](#) and is set with an instance of the [ElementSuite.Addin.RelayCommand](#) class. The [RelayCommand](#) class easily facilitates implementing the command pattern used by WPF and Element Suite to handle the action that occurs when a menu item is selected.
- The [WorkbenchTab](#) that is added to the instance of the [IWorkbenchService](#), is a subclass of [TabItem](#). This class extends [TabItem](#) to provide a close button on the displayed tab.
- [WorkbenchTab.Content](#) contains the visible content of the tab control and can be any WPF control or collection of controls you desire.

Distributed Assembly

- In Visual Studio, create a new Class Library project.
- Add a project reference to ElementSuite.Common.dll
 - This dll is located in the default Element Suite installation directory (%PROGRAMFILES(X86)%\5element\Element Suite\ by default).
- Create a class which implements [ElementSuite.Common.Interface.IWorkItem](#)
 - Annotate the class with the [System.SerializableAttribute](#)
 - Set the [IWorkItem.Id](#) property to a unique value per instance of the class
 - Define any other needed properties to encapsulate the work to be distributed
- Note that all properties of this class need to be defined with types which are annotated with [SerializableAttribute](#) so the object can be properly serialized.
- Create a class which implements [ElementSuite.Common.Interface.IWorkResult](#)
 - Annotate the class with the [System.SerializableAttribute](#)
 - Implement the [IWorkResult](#) properties with getters and setters
 - Define any other needed properties to preserve the state of the work result.
- Note that all properties of this class need to be defined with types which are annotated with [SerializableAttribute](#) so the object can be properly serialized.
- Create a class which implements [ElementSuite.Common.Interface.IWorkCommand](#)

- Define the [Execute](#) method to perform whatever task you want to be distributed using the two previous classes to retrieve the work to be executed and return the result.

Distributing The Work

- Call [IServiceLocator.Resolve<IWorkService>\(\)](#) to retrieve an instance of [IWorkService](#)
 - [IServiceLocator](#) is passed to the addin class as a parameter when [IAddin.Initialize](#) is called.
- Call [yourIWorkServiceInstance.CreateWorkQueue<FooWorkItem, FooWorkResult>\(\)](#) passing the types of the classes that were created in the distributed assembly which implement [IWorkItem](#) and [IWorkResult](#) and capture the [IAddinWorkQueue<FooWorkItem, FooWorkResult>](#) object that is returned.
- Call [IAddinWorkQueue<FooWorkItem, FooWorkResult>.Initialize<FooWorkCommand>\(\)](#) to initialize the work queue with the distributed assembly which will perform the execution of the distributed work.
- Add work items to the work queue using the [IAddinWorkQueue<FooWorkItem, FooWorkResult>.Enqueue](#) method and then mark the queue as completed by calling [IAddinWorkQueue<FooWorkItem, FooWorkResult>.CompleteEnqueuing](#).
- Finally invoke the [IAddinWorkQueue<FooWorkItem, FooWorkResult>.Start](#) method to begin processing the work.
- Retrieve the resulting work items from the [IAddinWorkQueue<FooWorkItem, FooWorkResult>.Results](#) property.
 - This property is of type [ReadOnlyObservableCollection](#) so the results can either be checked on a periodic basis or you can subscribe to change notifications on the Results property
 - [IAddinWorkQueue<FooWorkItem, FooWorkResult>.IsComplete](#) indicates when all results have been returned.

Index

Air, 3
Available Shared Services, 5
Creating Addin Assemblies
 Addin Assembly, 6
 Distributed Assembly, 6
Developing Addins, 5
Distributing The Work, 8
Earth, 3
Element Suite
 How To Install, 1
 What It Does, 1
 What It Is, 1
Figure 1.1, 4
Figure 1.2, 4
Figure 1.3, 7
Fire, 3
Pre-requisites and Tools, 6
Resources, 3
Troubleshooting/FAQs, 3
Water, 3