OpenSpan Article: Understanding Translators

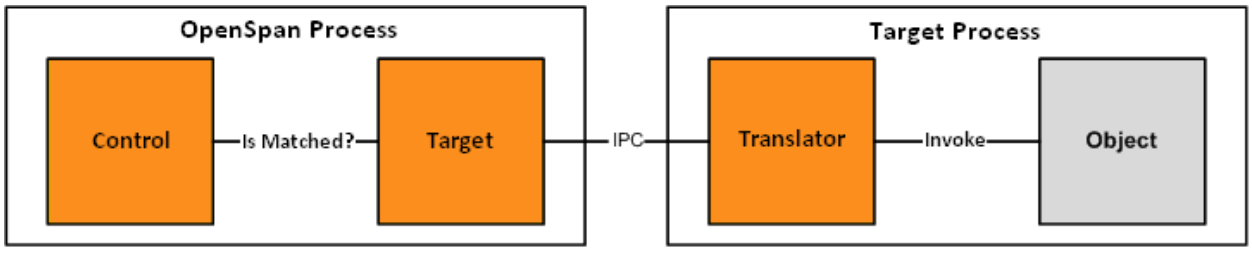
Published: March 31, 2008

What Translators Do

The OpenSpan platform allows developers to integrate applications at the desktop through their user interfaces. To integrate with an application, OpenSpan injects code within the target application that directly interacts with the application's objects. OpenSpan does not require the target application to be recompiled or modified in any way. OpenSpan supports a number of platforms and controls including Windows common controls, Internet Explorer HTML controls, Visual Basic, Java and .NET. However, in some cases, developers must create a special object, known as a *Translator*, to expose a user interface element to OpenSpan. Translators are appropriate in the following scenarios:

- The user interface element is a third-party component such as a grid or chart that OpenSpan does not yet support.
- The user interface element is a base platform control, but has been extended by the application developer to provide custom behavior.
- The application contains data that is not accessible via the user interface, but is available via properties or methods on a user interface element or a static class within the application.

In addition to the translator object, developers may be required to create additional classes to fully expose an object to OpenSpan. These classes, known as *controls* and *targets* exist within the OpenSpan process and are written in .NET. Controls and targets are necessary if the developer needs to expose properties, methods or events that do not exist in any common OpenSpan control. The diagram below illustrates the relationship amongst controls, targets, translators and application objects.

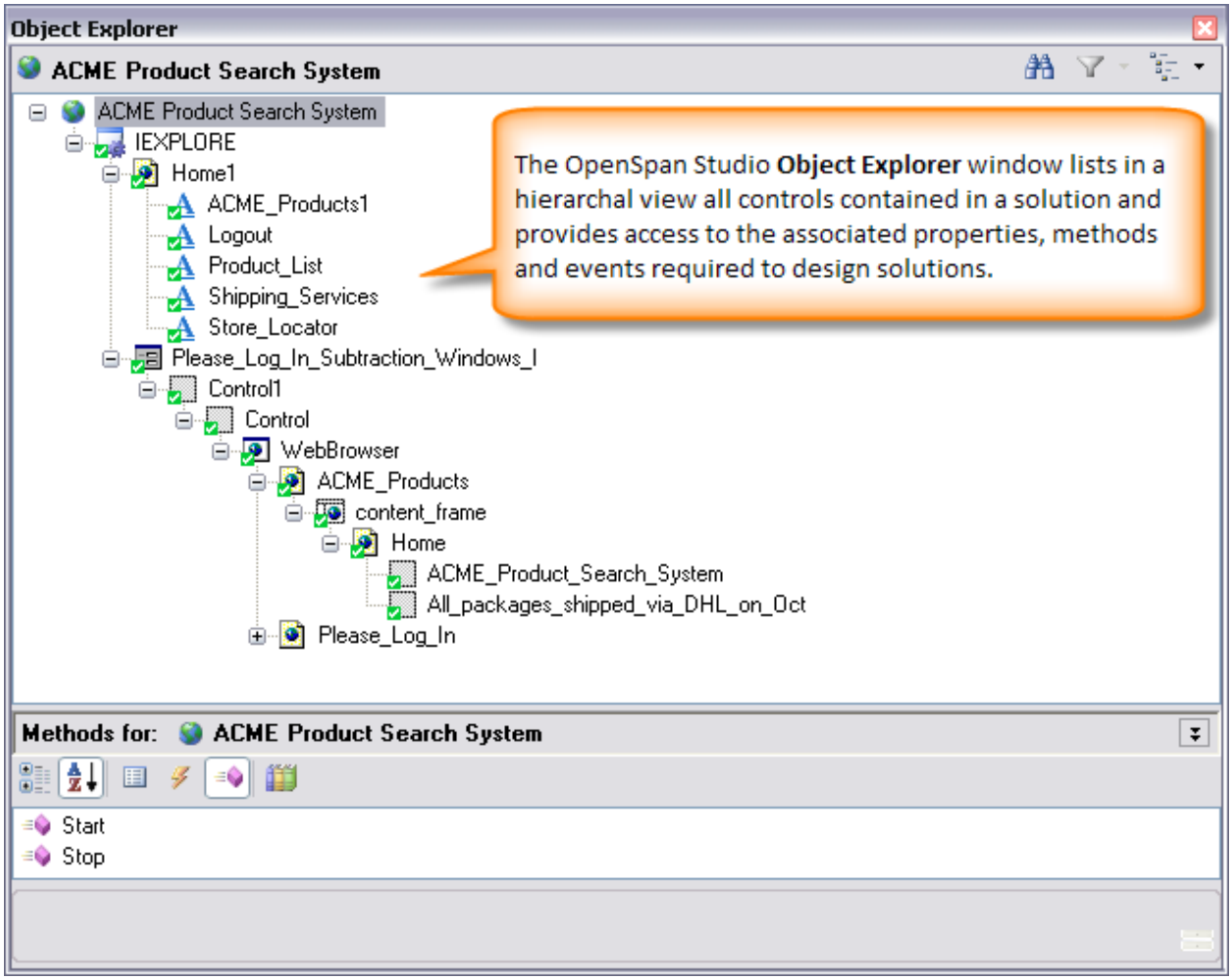


Controls

When an application object is interrogated in OpenSpan studio, a representation of that object is created. That representation is known as a *Control*. A control performs the following functions:

1. Saves the rules that are necessary to correctly identify the object the next time the application is run.
2. Acts as a proxy for the object so that it can be used in automations even when the user object has not been created within the target application.
3. Provides a consistent interface for the object regardless of the underlying application technology.

Controls are displayed within the OpenSpan Studio Object Explorer. A control is considered *created* when it has been matched, and *destroyed* when it has been unmatched.



Targets

A target is an OpenSpan object that corresponds directly with an application object. Whereas controls represent the *virtual* application object that does not change between runs of the application, targets represent the *actual* application object that exists within the application at any given moment. Targets are created and destroyed when application objects are created and destroyed. Targets are platform specific and encapsulate the necessary logic to automate and monitor application objects. Thus, the textbox control can interact with any target that implements the textbox interface, such as a windows textbox target, a Java textbox target or an HTML textbox target.

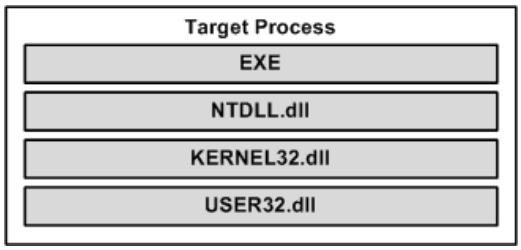
Translators

A translator is a unit of injected code that enables OpenSpan to automate and receive events from an application object. Targets communicate directly with translators using the OpenSpan IPC layer. Most of the time translators are objects, although in some cases translators can just be a single module or static class that interacts with all supported objects. Translators do not require any modifications or recompilation of the target application. OpenSpan automatically injects translators into a target application and attaches them to the appropriate application object. Translators are typically written in the same language as the objects they are interacting with. Thus for example, translators for .NET controls are written in C# and translators for Java controls are written in Java. The most notable exception to this is ActiveX or COM controls, which due to their binary interface can be written in C++ regardless of whether they were originally written in Visual Basic or another language.

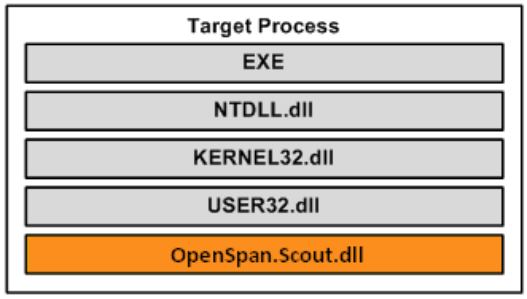
Translator Injection and Initialization Step by Step

The following step by step process describes how OpenSpan starts an application, detects a platform and loads translators.

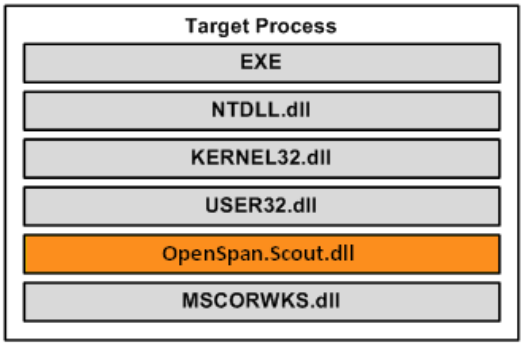
1. OpenSpan starts the target process. Windows allocates the memory for the target process and loads the executable and any statically linked modules including the base windows modules (NTDLL, KERNEL32, USER32, etc.).



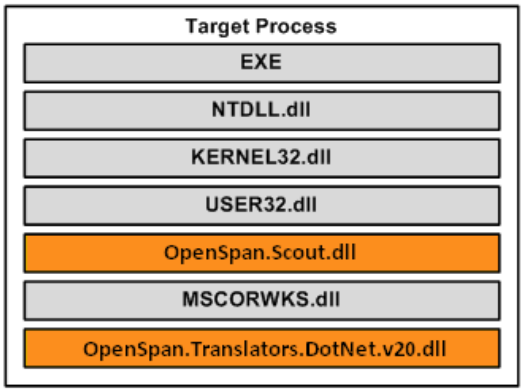
2. After the initial set of modules is loaded but before the application begins executing, OpenSpan injects the *Scout* module. The Scout module is responsible for hooking and monitoring all base Windows activity, including window creation and destruction, window sub-classing, module loading, and child process creation.



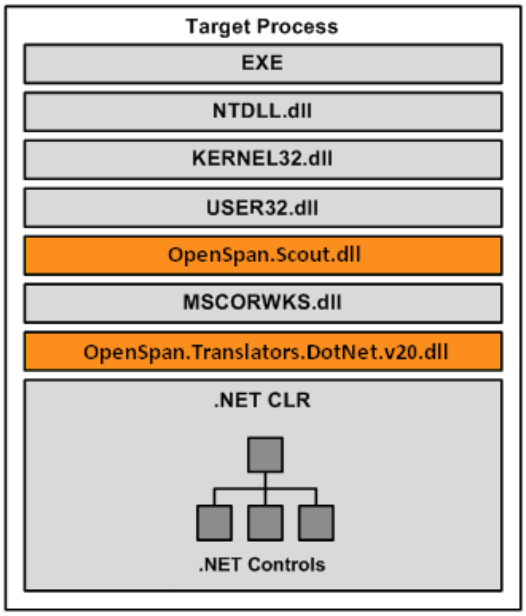
3. After Scout has been initialized, the application begins executing. During execution, Scout alerts OpenSpan each time a module is loaded.



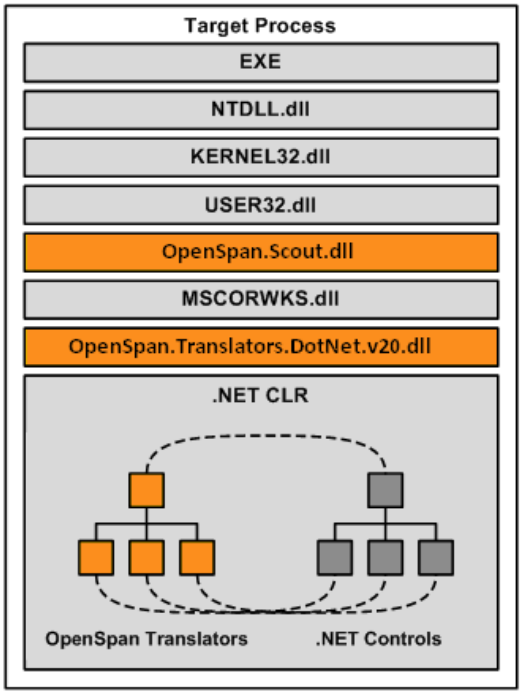
4. When a known module is detected, OpenSpan injects a translator module that is able to interact with the known module. In this case, after OpenSpan has detected the MSCORWKS.dll version 2.0 (which indicates that the .NET CLR is being loaded) OpenSpan injects the OpenSpan.Translators.DotNet.v20.dll. The translator then hooks a set of native CLR functions so that it can monitor CLR initialization.



5. When OpenSpan is notified via the hooked functions that the CLR is initialized, OpenSpan injects an additional .NET assembly to monitor the creation of .NET controls.



6. As .NET controls are created, OpenSpan creates a translator object to interact with each .NET control. The translator communicates with a corresponding OpenSpan control and target through the OpenSpan IPC layer.



Translator Benefits

The OpenSpan platform provides an extensible system to expose unsupported application objects quickly and easily using translator objects. Once a translator has been created, it can be used repeatedly within OpenSpan automations and adapters without additional updating or configuration.

More Resources

OpenSpan Studio Help documentation
For more information about Translators and additional tutorials, access material within OpenSpan Studio, by choosing **Help** | **Help Contents**, selecting the **Search** tab, and searching on the keyword *Translators.*

About the Author

Damon Lockwood founded OpenSpan and is currently its VP of Product Development.