Introduction

This tutorial shows you how to call the eBay Finding service using the FindingKit provided by eBay. It also shows you how to update the FindingKit to a newer version of the Finding service when it becomes available.

FindingKit provides a simplified client interface which wraps around the standard .Net WCF proxy for eBay Finding service. The kit makes it easy for a developer to interact with the eBay Finding service by hiding common tasks like xml serialization and HTTP header settings and by providing common facilities like message logging.

This tutorial shows you how to access the eBay Finding service using the FindingKit. In this tutorial, you'll build a simple console application that displays the finding result, using the findItemsByKeywords call.

Complete Source Code

The complete code is provided in the FindingKit package Samples\ConsoleFindItems subdirectory.

Before You Begin

There are a few prerequisites for completing this tutorial:

1. You must have installed Visual Studio 2008 or above.
2. You must have joined the eBay Developers Program and obtained application keys for the eBay production or sandbox environment.

Step 1: Create the Project in Visual Studio

Create a Visual Studio solution and add a new console application called ConsoleFindItems in the solution (see Fig. 1).

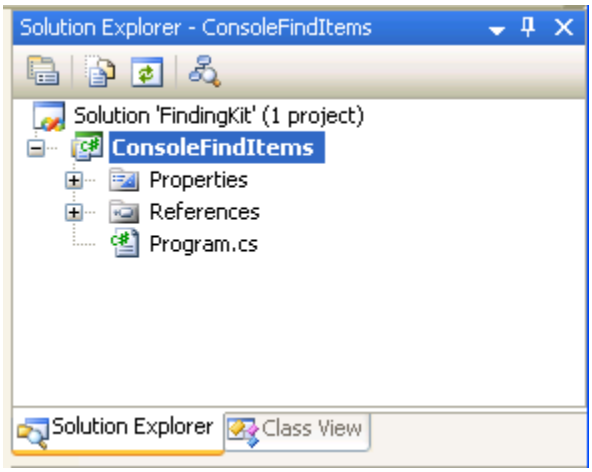

Fig 1. Create a ConsoleFindItems Console Application

Step 2: Add Reference

Add into the solution the eBay.Services project, which you can find in the FindingKit\eBay.Services folder, and then add it as a project reference to the ConsoleFindItems project. You will see that the eBay.Services assembly is referenced by ConsoleFindItems project (see Fig 2). The eBay.Services project contains .Net WCF proxy for eBay Finding service and a few helper classes to simplify interactions with eBay Finding service.

Also add SLF (Simple Logging Facade) assembly reference to the ConsoleFindItems project (see Fig 2). You can find SFL assembly in the FindingKit\Lib\SFL folder. SLF is used by eBay.Serivces for message logging; in the sample, we will also use SLF to output result to console window.

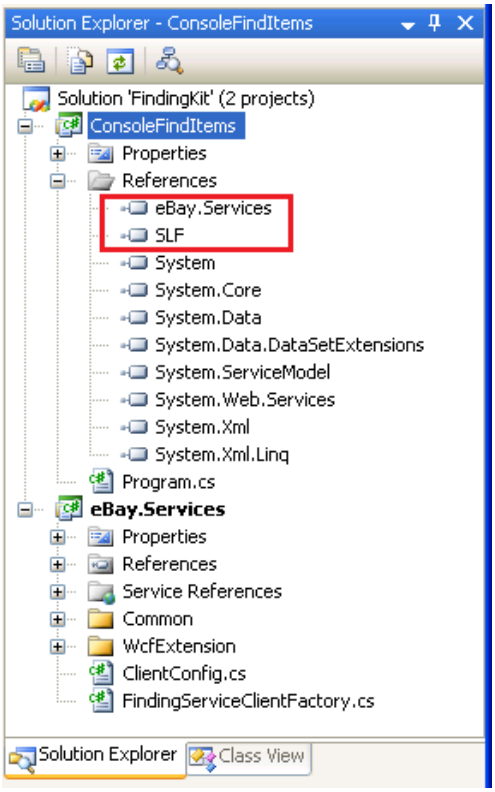

Fig 2. ConsoleFindItems References

Step 3 Create the Main Program

Add the following code in the Program.cs file (see Listing 1):

```
using System;
using eBay.Services;
using eBay.Services.Finding;
using Slf;

namespace ConsoleFindItems
{
    /// <summary>
    /// A sample to show eBay Finding service call using the simplified interface
    /// provided by the FindingKit.
    /// </summary>
    class Program
    {
        static void Main(string[] args)
        {
            // Init log
            LoggerService.SetLogger(new ConsoleLogger());
            ILogger logger = LoggerService.GetLogger();
```

```
            try
            {
                // Initialize client site configuration
                ClientConfig config = new ClientConfig();
                // Initialize service end-point configration
                config.EndPointAddress = "http://svcs.ebay.com/services/search/FindingService/v1";
                // set eBay developer accoutn AppID
                config.ApplicationId = "You AppID here";

                // Create a service client
                FindingServicePortTypeClient client =
                                    FindingServiceClientFactory.getServiceClient(config);

                // Create request object
                FindItemsByKeywordsRequest request = new FindItemsByKeywordsRequest();
                // Set request parameters
                request.keywords = "harry potter phoenix";
                PaginationInput pi = new PaginationInput();
                pi.entriesPerPage = 2;
                pi.entriesPerPageSpecified = true;
                request.paginationInput = pi;

                // Call the service
                FindItemsByKeywordsResponse response = client.findItemsByKeywords(request);

                // Show output
                logger.Info("Ack = " + response.ack);
                logger.Info("Find " + response.searchResult.count + " items.");
                SearchItem[] items = response.searchResult.item;
                for (int i = 0; i < items.Length; i++)
                {
                    logger.Info(items[i].title);
                }
            }
            catch (Exception ex)
            {
                // Handle exception if any
                logger.Error(ex);
            }

            Console.WriteLine("Press any key to close the program.");
            Console.ReadKey();

        }
    }
}
```

Listing 1. Main Program

The program starts by importing .Net and FindingKit namespaces. Here we import:

1. eBay.Service for FindingServiceClientFactory and ClientConfig classes reference
2. eBay.Service.Finding for Finding service types reference
3. SLF for logging

The Main function shows a typical Finding service call flow.

1. Initialize Logging

FindingKit internally uses SLF for HTTP messages and SOAP payload logging, so we need to initialize logging first if we want to enable message logging. For detailed information about SLF details, for example, how many log levels are supported, how to log to a file, etc., please refer to a good introduction here.

2. Set Up Client Configuration

Client-side configuration, such as service endpoint address and application id, must be set up before the client can communicate with eBay services. In the sample, we instantiate a ClientConfig instance and set configurations accordingly. For all supported configurations, please refer to the source of the ClientConfig class. Note that some configurations are mandatory, such as service endpoint address and application id, while others are optional, such as enabling logging.

3. Create Service Client

An applications use the service client to communicate with a service. In the sample, we can easily get a Finding service client instance by invoking the static factory method on the FindingServiceClientFactory class.

4. Create Outbound Request and Setup Request Parameters

To call an operation on a service, you need to create a request and populate the request parameter. In the sample, we create a FindItemsByKeywordsRequest instance, and populate keyword and pagination information accordingly. For all supported input parameters of a request type, please refer to eBay Finding Service Call Reference.

5. Call the Operation on the Service Client and Receive Inbound Response

Real interaction between an application and an eBay service takes place here, where you call an operation on the service client, pass in the request instance as parameter. If the call is successful, you will get a corresponding response instance. Behind the scene, .Net WCF proxy will do the low level message communication tasks. In the sample, we call findItemsByKeywords operation on the client, and pass in the FindItemsByKeywordsRequest instance we created in the previous step. On a successful call, we will get a FindItemsByKeywordsResponse instance.

6. Handle Response

Once you get the response, it's up to you to decide how to further handle or present the response. In the sample, we simply log the found items numbers and titles to the console window. For all supported output parameters of a response type, please refer to eBay Finding Service Call Reference.

7. Handle Exception

If any of the above steps goes wrong (for example, if the service call fails and throws an exception), it's the application's responsibility to capture and handle exceptions. In the sample, we simply log the exception to the console window.

Step 4 Run the Application

Before running the program, you must substitute your own eBay Developer Account Application ID in the code, and then rebuild the application using Visual Studio 2008 or above and run it.

If everything works fine, you will see result similar to Fig. 3.

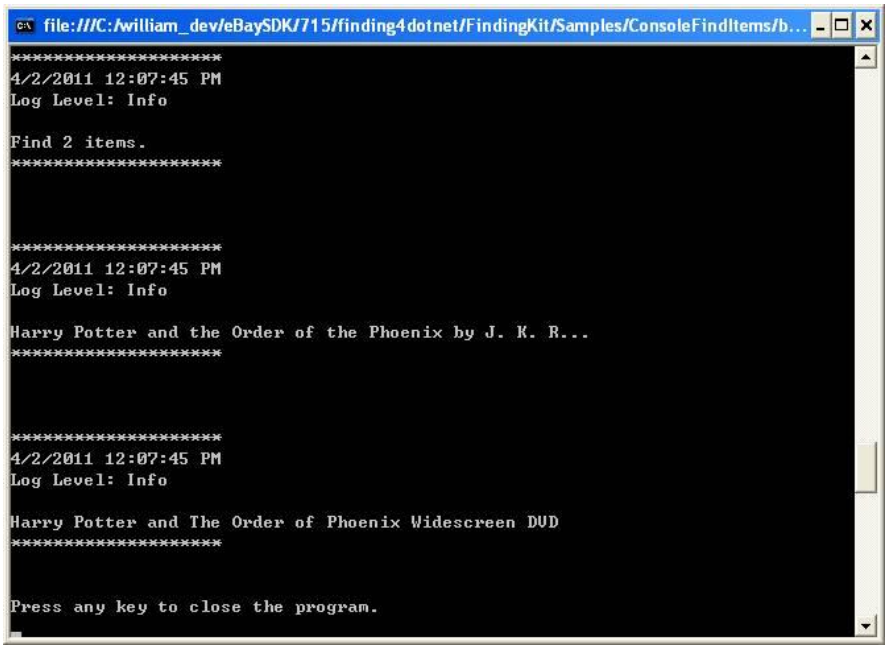

Fig. 3. Console Output

Now you have a working sample that can call eBay Finding service via FindingKit. Congratulations!

How to Update the FindingKit to a New WSDL

FindingKit has reference to the latest eBay Finding wsdl here:

http://developer.ebay.com/webservices/Finding/latest/FindingService.wsdl

The current WSDL version (at the time of the FindingKit package build) is 1.9.0.

If a new Finding service version is published by eBay, you can simply update (or sync) to the latest version (see Fig. 4):

1. Open the 'Service References' folder in the eBay.Services project.
2. Right-click the 'Finding' service.
3. In the popup menu, click 'Update Service Reference'.

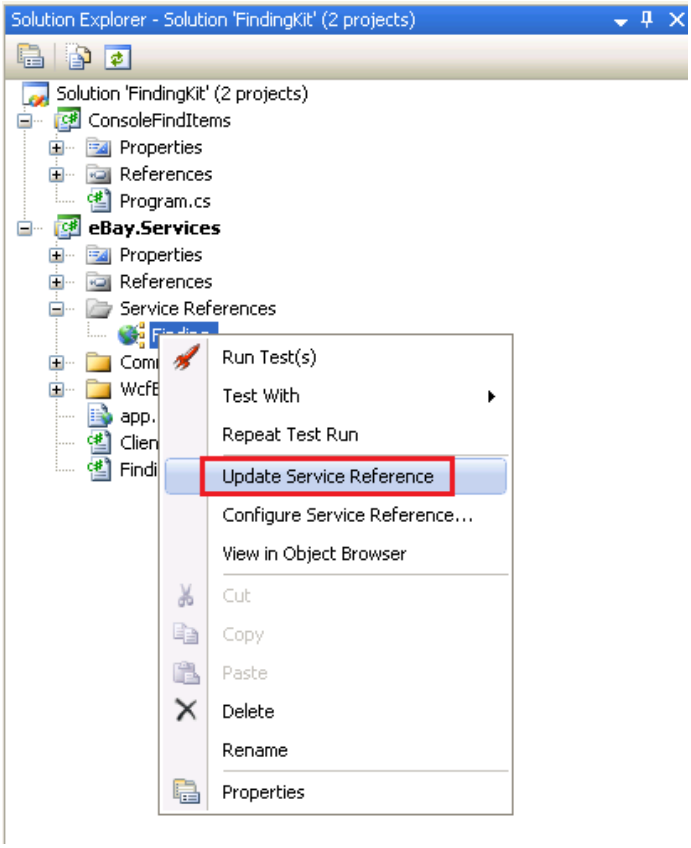

Fig. 4. Update Service Reference

Then your FindingKit is updated.

If for some reason, you need to update to a specific version of the eBay Finding service, for example, 1.8.0, just perform the following steps (see Fig. 5):

1. Open the 'Service References' folder in the eBay.Service project.
2. Right click the 'Finding' service.
3. In the popup menu, click the 'Configure Service Reference'.
4. In the shown dialog, replace the wsdl address with the new one, for example: http://developer.ebay.com/webservices/Finding/1.8.0/FindingService.wsdl
5. Click OK and wait until the update process finishes.

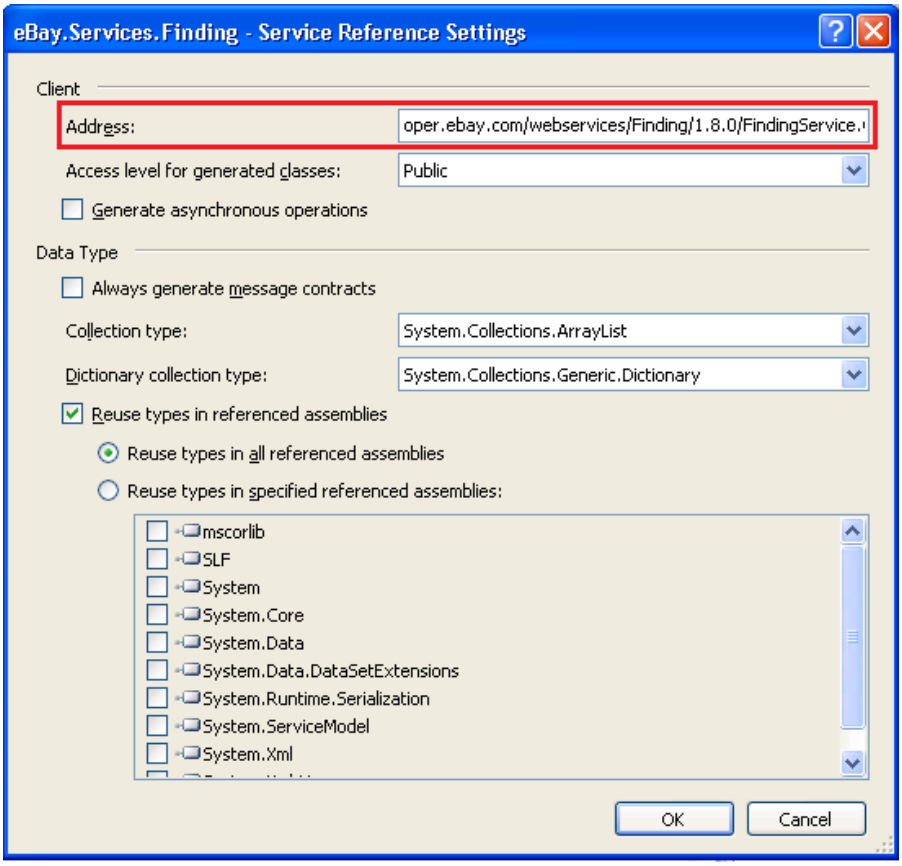

Fig. 5. Configure Service Reference