e-Learning

SkillSoft
AHEAD OF THE LEARNING CURVE

OLSA Client Toolkit Readme

www.skillsoft.com

Copyrights and Trademarks

Copyrights

Printed in the United States of America
SkillSoft Corporation
107 Northeastern Blvd.
Nashua, NH 03062
603-324-3000
87-SkillSoft (877-545-5763)
Information@SkillSoft.com

Trademarks

"Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries."

Dreamweaver® is a registered trademark of Macromedia, Inc. in the United States and/or other countries.

Adobe and PhotoShop® are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Sound Forge ® Audio Studio™ and Sound Forge ® 7.0are trademarks or registered trademarks of Sony Pictures Digital Inc. or its affiliates in the United States and other countries."

Mozilla Firefox™ and the Firefox logo are trademarks of The Mozilla Foundation.

All trademarks appearing on the Netscape Network are the property of their respective owners, including, in some instances, the Netscape Network and its affiliates, including, without limitation, Netscape Communications Corporation and America Online, Inc.

The term "Linux" is a registered trademark of Linus Torvalds, the original author of the Linux kernel.
"Sun, Sun Microsystems, Sun JVM/JRE, logos, are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries."

All other names and trademarks are the property of their respective owners.
Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including Photocopying or recording, for any purpose without the express written permission of SkillSoft Corporation.
This document is provided for information only. SkillSoft makes no warranties of any kind regarding the SkillSoft software, except as set forth in the license agreement. The SkillSoft software is the exclusive property of SkillSoft and is protected by United States and International copyright laws. Use of the software is subject to the terms and conditions set out in the accompanying license agreement. Installing the software signifies your agreement to the terms of the license agreement.

Table of Contents

INTRODUCTION

The OLSA Integration kit contains files and documentation necessary to set up the Client-side of an OLSA Web Services implementation. This document is a guide to the information and materials included in the integration kit and provide assistance to system engineers responsible for configuring/testing their LMS or portal to work with OLSA.

SYSTEM REQUIREMENTS

Java Implementation

- JDK 1.4.2
- Apache ANT

.net Implementation

- Visual Studio Environment
- Web Service Security Toolkit WSE 2.0

For Microsoft WSE 3.0 support with, .NET 2.0 and .NET 1.1, make the following changes:

- Update the C# files within the client toolkit that currently import the WSE 2.0, instead of having the following using statements:

```
using Microsoft.Web.Services2.Security;
using Microsoft.Web.Services2.Security.Tokens;

…they will need to have…

using Microsoft.Web.Services3.Security;
using Microsoft.Web.Services3.Security.Tokens;
```

You could also use preprocessor directives to accomplish this:

```
//define the version of WSE you wish to use here
//for the example, we are using WSE 3.0
#define WSE3
#if (WSE3)
      //if WSE3 is used then import the libraries
      using Microsoft.Web.Services3.Security;
      using Microsoft.Web.Services3.Security.Tokens;
#elif (WSE4)
      //WSE 4.0 does not exist. This is an example of how
      //you could include future versions.
#else
      //use WSE2 as the default option
      using Microsoft.Web.Services2.Security;
      using Microsoft.Web.Services2.Security.Tokens;
#endif
```

- In the OlsaProxy.cs class, you must change the base class to use WSE 3.0. To do this, change the line:

```
public class OlsaService : Microsoft.Web.Services2.WebServicesClientProtocol
```

to

```
public class OlsaService : Microsoft.Web.Services3.WebServicesClientProtocol
```

- Update the following within the OlsaProxy.cs class to point to the endpoint SkillSoft issued you:

```
public OlsaService()
{
      this.Url = "http://localhost:8083/olsa/services/Olsa";
}
```

Paste the SkillSoft issued endpoint in place of 'http://localhost:8083/olsa/services/Olsa'.

DOCUMENTATION

Several supporting documents are available to support the OLSA Integration effort:

- **OLSA Integration Kit Readme** – (this document) A guide to the information and materials included in the integration kit.
- **OLSA Integration Guide** - provides detailed information on the web services and functions that make up the OLSA.
- **OLSA Release Notes** – details on new features, changes, bug fixes, and known issues with each release of OLSA.

INTEGRATION KIT COMPONENTS

Root Directory	Sub-Directories	Notes
ROOT		▪ build.xml and build.properties - Files used to do the build ▪ Client-toolkit-exepected-output.txt – sample file shows expected output when running the full build
\config		▪ client_deploy.wsdd ▪ olsa_user.properties - Change to point to right endpoint and customer ID etc.
\docs		▪ OLSA Integration Kit Implementation Guide -This document. ▪ olsa.wsdl - WSDL specification for the current OLSA release. ▪ OLSA WEB Services Definition.doc - Details of OLSA Web Services. This is a full description of all OLSA Web Services, including those not part of this kit. ▪ OLSA Release Notes - Limitations and known issues. ▪ OLSA Search XSD Documentation – A description of the elements and attributes that define the XML output of the Search service.
\dotnet	\AssetIntegration \Assignement \Configuration \OfflineIntegration \SearchandLearn \SignOnService \UsageData \UserManagement \Utility	▪ Folders contain examples of .net code for the different OLSA functions ▪ Each folder contains the .net file set to use the examples ▪ Utility folder – Contains files compiled as a .dll used by the .net examples. The folder also contains files used by the OLSA Utility Service.
\lib		▪ Libraries used in building JAVA examples. Put it in the classpath.
\src	\example \assetintegration \assignment \configuration \offlineintegration \searchandlearn \signon \usagedatasynchronization \usermanagement \uitility	▪ Folders contains all the JAVA example source code and some Axis generated stubs to communicate to OLSA. ▪ OlsaServiceTest.java - This is a simple code example that runs basic tests. ▪ Utility folder – files used by the OLSA Utility Service.

SET-UP

olsa_user.properties.

Before running any example code make sure to set the correct values in the \config\olsa_user.properties file.

1. Setup the endpoint. The endpoint must point to valid OLSA server (provided by SkillSoft). Make sure the port number is correct. Example:

   ```
   endpoint=http://localhost:8083/olsa/services/Olsa
   ```

2. Ensure shared secret key is correct (provided by SkillSoft). Example:

   ```
   sharedsecrect= TVk1dNuSrOmO2xN
   ```

3. Ensure customerid is valid (provided by SkillSoft). Example:

   ```
   customerid=sp63501c
   ```

CLIENT EXAMPLES

The integration kit contains code samples that you can use to test OLSA communications. The file Client-toolkit-exepected-output.txt in the root folder of the integration kit shows an example of what the returned output should look like when the communications test is succesful.

Build and run test code (JAVA).

Use Ant to run the build, compile client code, and run the JAVA examples. You can run the test from the command console or through a third-party development application.

Build targets:

- build - builds and runs example code
- compile - compiles client code
- run-all - Runs all example code

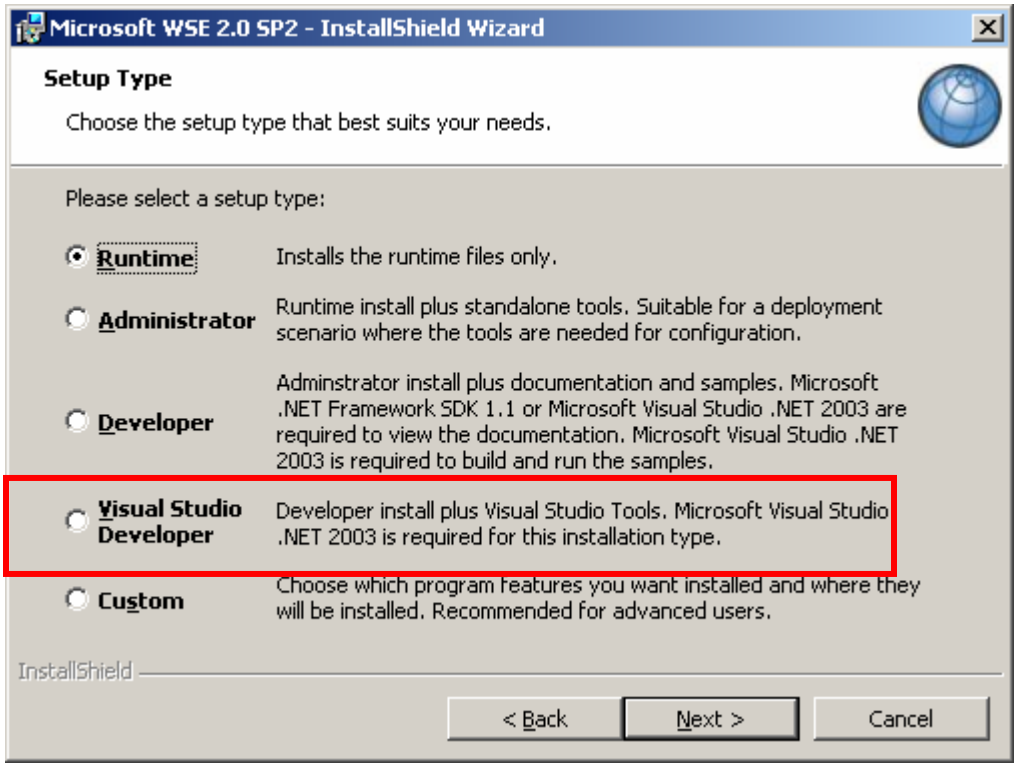
Build and run test code (.net)

Use Visual Studio to run the build, compile client code, and run .net examples.

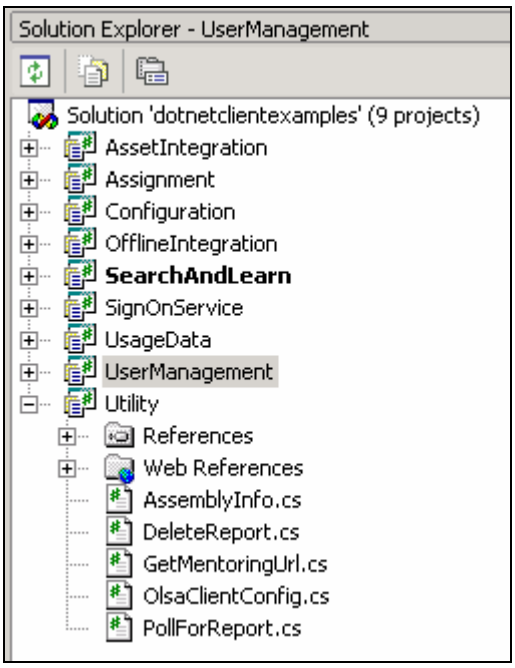
1. Set up Visual Studio

   - Install Visual Studio Environment 2003 for C# or your language of choice

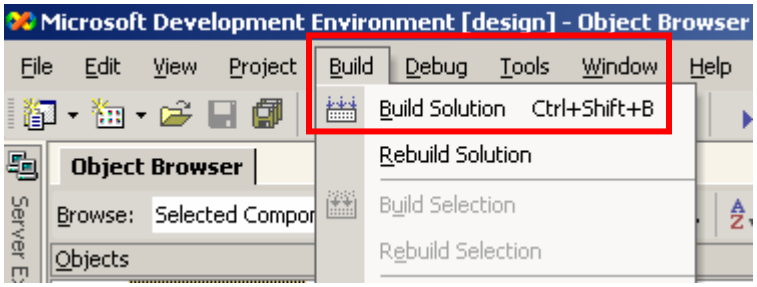
   - Install Web Service Security Toolkit WSE 2.0. Select the Visual Studio Developer option when installing WSE



2. Launch Visual Studio .NET and open dotnet\dotnetclientexamples.sln. You will see nine projects in the Solution Explorer:

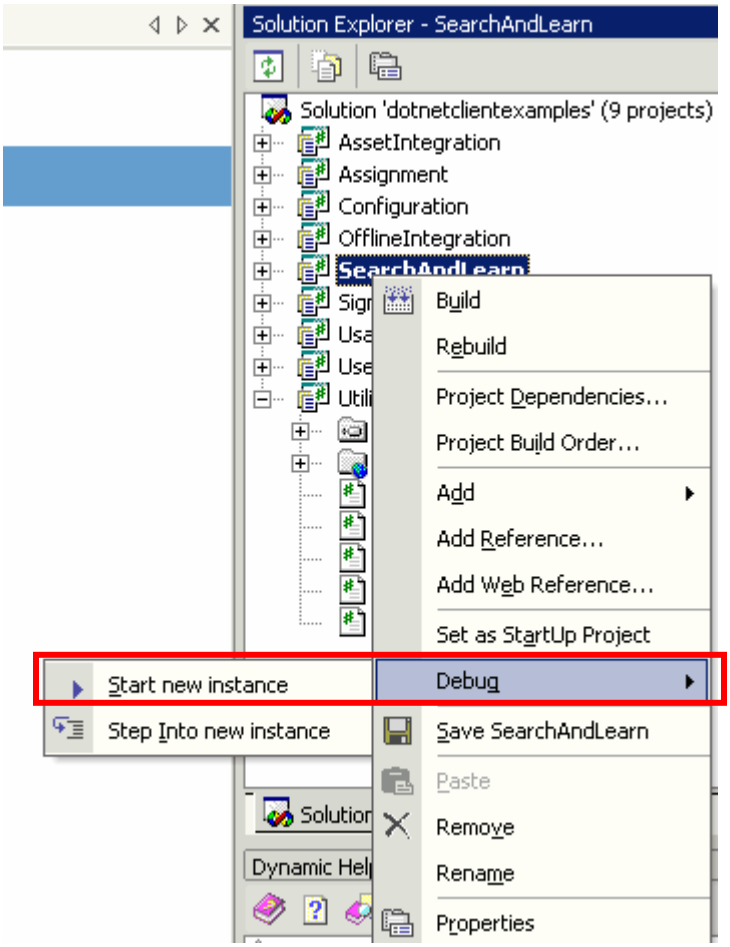


3. Choose Build to build the solution.



4. Open **olsa_user.xml** and update the endpoint, customerid and sharedsecret as described previously.

5. Right-click on a project choose **debug**



This will launch the project and open a console.

You can also launch the executable from the command line. Open a command console and Browse to you clienttoolkit\dotnet\AckResetExample\bin\Debug and run AckResetExample. You can pass the endpoint, customerid, and sharedsecret from command line as well.

Code Samples

The OLSA Integration kit contains example code for both JAVA and .net implementations. A full description of the OLSA functions executed in the examples is included in the **OLSA Integration Guide**.

BEST PRACTICES FOR PROGRAMMING ASSET INTEGRATION SERVICES

When using the OLSA Asset Integration Service for catalog synchronization please ensure that you adhere to the following coding best practice to handle specific exceptional circumstances.

Background

The following example illustrates at a high level how your LMS should execute the OLSA Asset Integration Service Initiate-Poll-Acknowledge cycle:

```
  handle = AI_InitiateAssetMetaData (mode=all or delta)
  while (true) {
     Sleep for N minutes
     AI_PollForAssetMetadata(handle)
     If (url-returned) {
        break;
     }
  }
  if (url_returned) {
    Retrieve zip file via URL
    // course-processing-loop A
    while (there-are-courses-to-process-in-the-zip-file)
{
                if (OLSA-course-status-is-entitled) install the course;
                if (OLSA-course-status-is-not_entitled) uninstall the course;
                if (OLSA-course-status-is-modified) update the course;
   }
   AI_AcknowledgeAssetMetadata(handle)
  }
```

Exceptional Circumstance

here is a remote possibility that OLSA will return false positives for entitled, not_entitled and modified status values for content under the following scenario.

During the AI_AcknowledgeAssetMetadata() call, OLSA crashes and cannot process the request. If this occurs then on the next iteration of the Initiate-Poll-Acknowledge cycle, the LMS will receive the same set of changes as in the previous cycle (plus additional changes that might have occurred in the interim).

Depending on the nature of the Acknowledge failure, the caller could see an OLSA GeneralFault or some sort of network exception. There really is no recourse for the LMS at this point. Nor should there be a need for recourse since the LMS has successfully processed its side of this cycle.

Recommendation

SkillSoft recommends coding the course processing loop defensively to make this exceptional circumstance transparent to your LMS. The LMS should first verify if the course is installed or uninstalled, then perform the respective install or uninstall action as shown in the following example:

```
        // course-processing-loop-B
    while (there-are-courses-to-process-in-the-zip-file) {
                if (OLSA-course-status-is-entitled) {
                     if (course-is-uninstalled-in-LMS) install the course;
                }
                if (OLSA-course-status-is-not_entitled) {
                     if (course-is-installed-in-LMS) uninstall the course;
                }
                if (OLSA-course-status-is-modified) update the course;
   }
```

If an a priori check is not possible, then SkillSoft recommends wrapping the install and uninstall actions in an exception-handling block so that the LMS can ignore the relevant exception that could occur as shown in the following example:

```
        // course-processing-loop-C
    while (there-are-courses-to-process-in-the-zip-file) {
                if (OLSA-course-status-is-entitled) {
                     try {
                       install the course;
                     } catch (AlreadyInstalledException) {
                        //ignore
                     }
                }
                if (OLSA-course-status-is-not_entitled) {
                     try {
                       uninstall the course;
                     } catch (NotInstalledException) {
                        //ignore
                     }
                }
                if (OLSA-course-status-is-modified) update the course;
   }
```

The update-action is an idempotent operation, which means the operation has no real impact, so you can perform the update redundantly without any special checking or handling.