

AICC Watch™ Documentation

Contents

Purpose	1
System Requirements	1
Installing Internet Information Services (IIS)	1
Creating the Virtual Directory	2
Running AICC Watch	3
AICC Watch Features	4
AU Settings.....	4
Window Settings	5
Read-Only Settings.....	5
Read/Write Settings.....	5
GetParam Messages	6
PutParam Messages.....	7
PutInteractions Messages.....	7
PutObjectives Messages	8
PutPerformance Messages	8
ExitAU Messages.....	8
Most Recent Command	8
Advanced Uses of AICC Watch.....	9
Architecture	9
Running AICC Watch from Visual Studio.....	9
Example: Debugging with Flash Builder and Visual Studio	10

Purpose

We created *AICC Watch* to provide a test bed while we added AICC support to our *Training Studio*[®] and *Exam Engine*[™] products. While there is a free test suite (and lots of good documentation) available at <http://www.aicc.org>, we found it difficult to use on Vista/Windows 7 because it created these strange modal dialog boxes that kept you from running the Learning Management System (LMS) simulation in one window and your content in another. The AICC Test Suite also requires you to install the Apache web server. That was not a big problem but it takes time and effort to get it running alongside Internet Information Services (IIS).

For AICC Watch, we followed the model of our popular *SCORM Watch*[™] product to make an interactive LMS simulation that you can use to make sure your content sends and receives bookmarks, interaction information, and other information. This document explains how to configure and use AICC Watch.

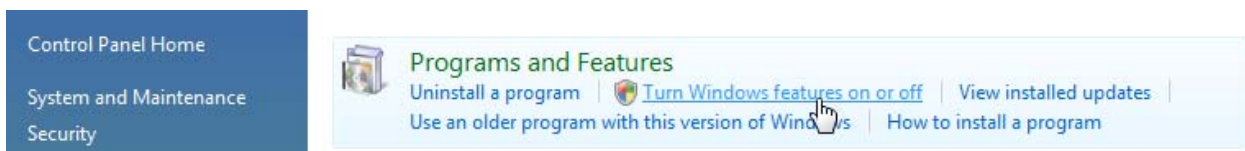
While there are several flavors of AICC, this product deals with the HACP (HTTP AICC CMI Protocol) interface. This is where the content communicates with the LMS using HTTP Post.

System Requirements

AICC Watch is an ASP.NET 4.0 application. That means that the computer running the product must have IIS installed and the free .NET 4.0 runtime. The installation will configure the virtual directory and add .NET 4.0 if it is not already there, but you must have IIS installed first.

Installing Internet Information Services (IIS)

If you don't already have IIS installed, you add it by *Turning Windows features on or off* as shown below.



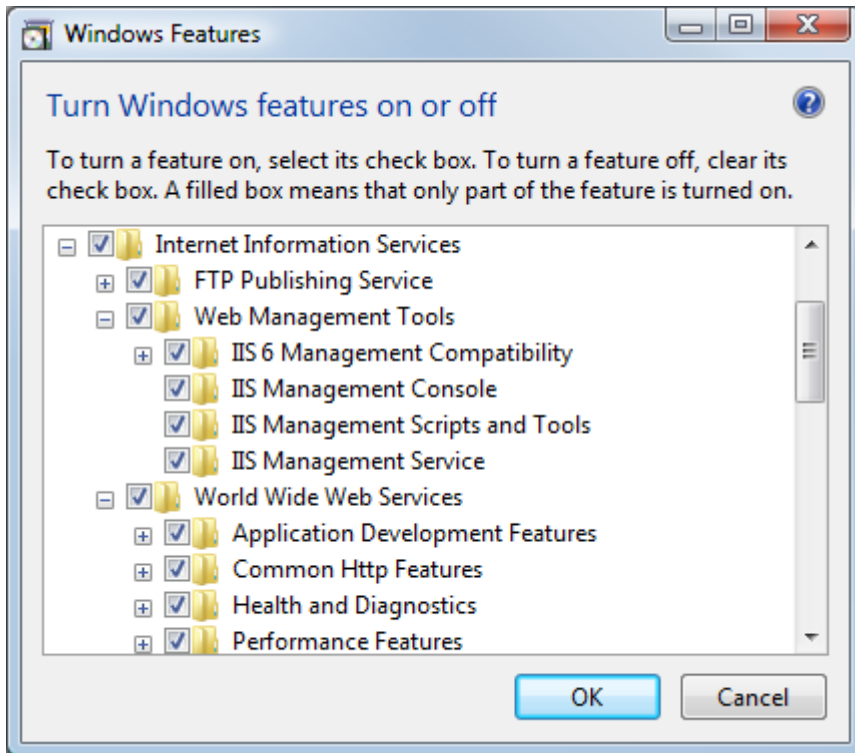
Then make sure that Internet Information Services and its various components are checked as shown below. The steps vary slightly by operating system. Here are some links:

Windows XP:

<http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/iiiisin2.msp?mfr=true>

Vista and Windows 7:

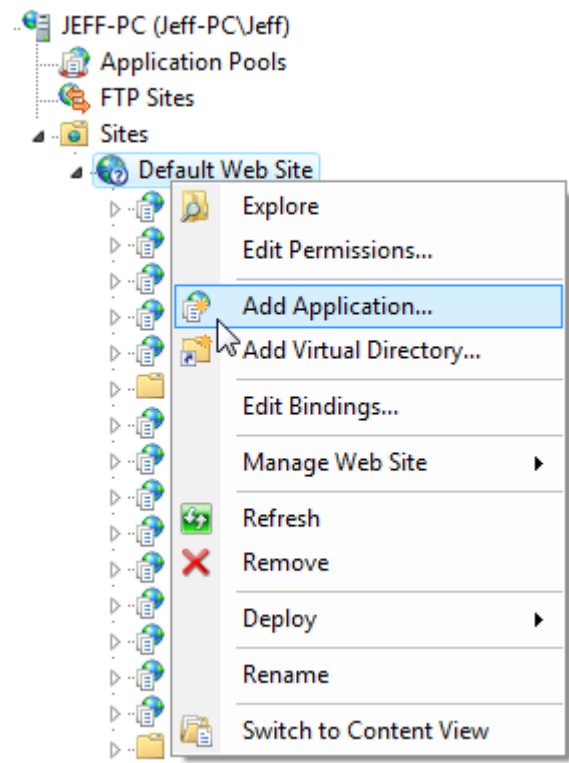
<http://learn.iis.net/page.aspx/28/installing-iis-7-on-windows-vista-and-windows-7/>

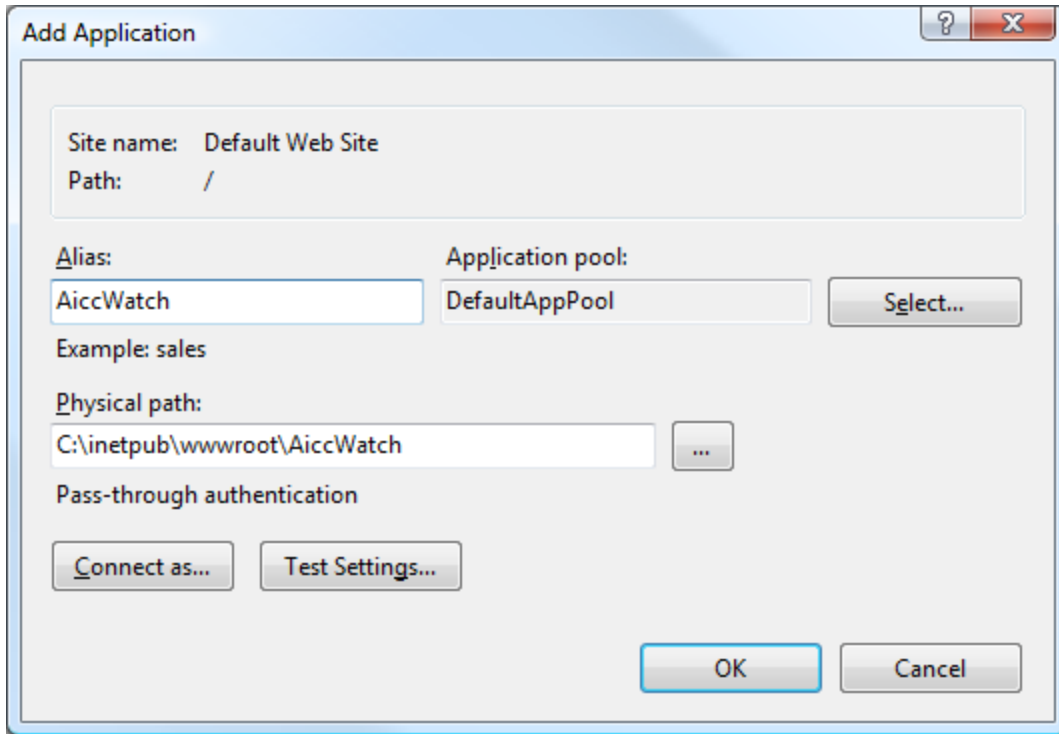


Creating the Virtual Directory

If you need to create the virtual directory yourself, follow these steps.

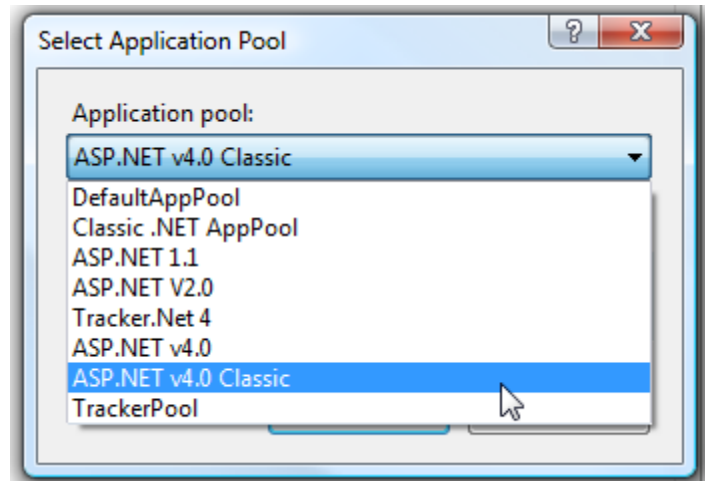
1. Locate the location of the AICC Watch files. By default, these will be at C:\inetpub\wwwroot\AiccWatch.
2. Launch IIS Manager.
3. Right-click the *Default Web Site* and select *Add Application...* as shown the right.
4. Enter your *Alias* (typically "AiccWatch" but can be another name) and navigate to the appropriate *Physical path* (such as "C:\inetpub\wwwroot\AiccWatch").





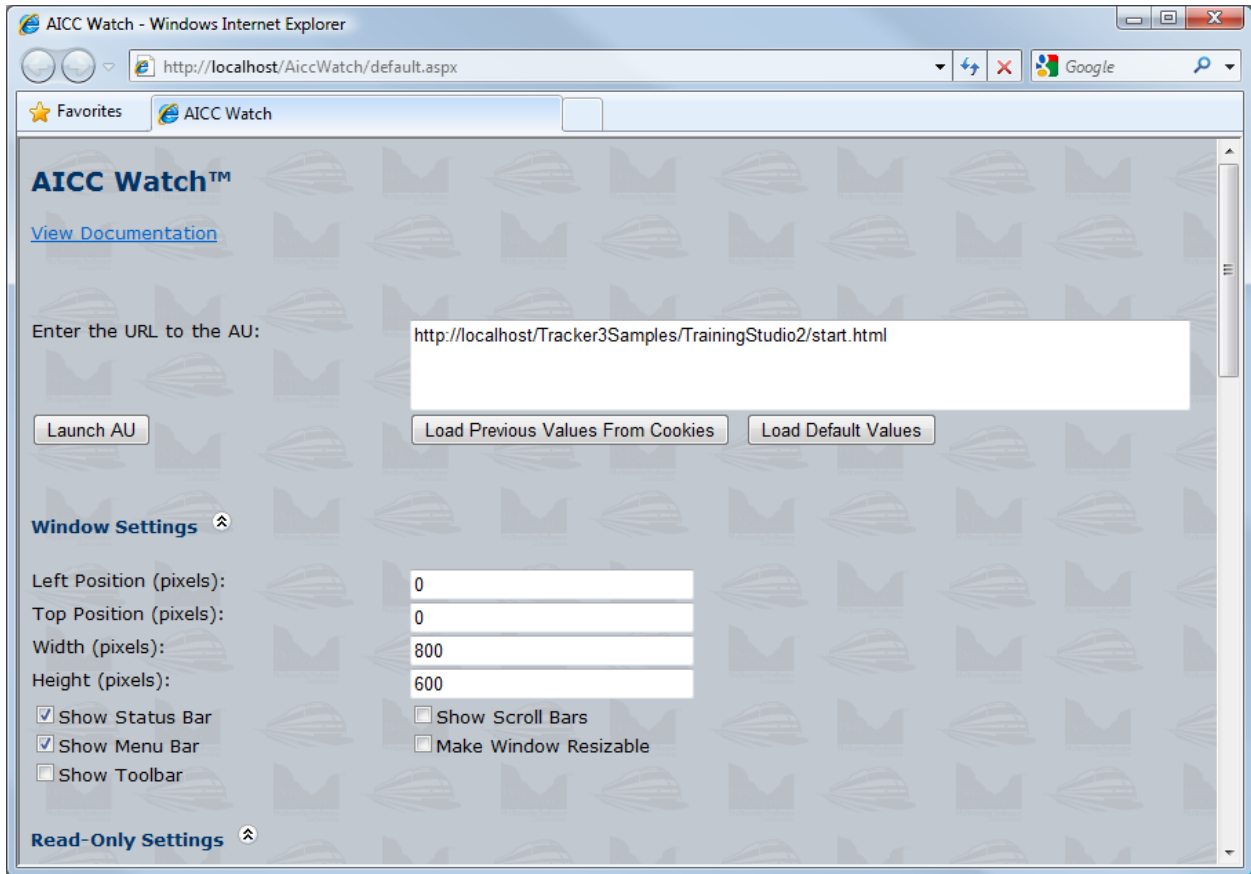
5. Click the *Select...* button to set the *Application pool*. You want this to be ASP.NET v4.0 Classic as shown to the right. You can also set it to *ASP.NET v4.0* if desired.

The above steps are for IIS 7. If you are using IIS 6, the steps are similar except that the ASP.NET version is set on the *ASP.NET* tab of the Properties dialog for the application. Please contact Platte Canyon support if you have any questions or problems.





Running AICC Watch

You can launch AICC Watch from the *Start Menu – Programs – AICC Watch*. Or you can type localhost/AiccWatch into your browser. The result is shown below.



AICC Watch Features

AICC Watch is broken into different areas. Most of them can be collapsed by clicking the  graphic or expanded by clicking the  graphic. We will go through each screen area in turn.

AU Settings

Enter the URL to your Assignable Unit (AU) lesson. If it is HTML, then you typically must be in the same domain (e.g., *localhost*) for security reasons. If it is Flash or Silverlight content, you can link to a different domain since AICC Watch has the appropriate *clientaccesspolicy.xml* and *crossdomain.xml* files to allow access. This URL is stored in a cookie and automatically populated the next time you start the program.

When you are ready to launch the AU, click the *Launch AU* button. This will launch the content in a separate window described in the *Window Settings* section. If you have two monitors, you might make the AU window display on your other monitor.

Click the *Load Previous Values From Cookies* button to put AICC watch in the same status the last time you ran an AU. You might want to do this to test bookmarking, audio settings, etc.

Click the *Load Default Values* button to put all the various value in their default state.

Window Settings

The settings in this section control the left, top, width, and height of the window. You can also set the characteristics of the window such as whether it has a status bar, a menu bar, a toolbar, and scroll bars. You can also make the window resizable.

Read-Only Settings

This section is for items that are passed *from* AICC Watch *to* your lesson. This include the *AICC Version*, an optional *AU Password* (which must match that of the lesson if used), a *Student ID*, the *Student Name*, the *Mastery Score* (between 0 and 100), a *Credit Value*, the *Lesson Mode*, the *Time Limit Action*, the *Maximum Time Allowed*, the *Session ID*, and *Launch (Core Vendor) Data*. For those items with defined values (*Credit Value*, *Lesson Mode*, and *Time Limit Action*), you can select values from a drop-down list. AICC Watch automatically creates a GUID (Globally Unique Identifier) for the *Session ID*, but you can enter your own value if desired. This section is shown below.

Read-Only Settings	
AICC Version:	3.0
AU Password (optional):	
Student ID:	44
Student Name (Last, First MI):	Rhodes, Jeffrey
Mastery Score (Decimal):	80
Credit Value:	Credit
Lesson Mode:	normal
Time Limit Action:	continue,no message
Maximum Time Allowed (blank or HH:MM:SS):	
Session ID:	efb03079-4f90-42c9-a56c-8e74f75c07fb
Launch (Core Vendor) Data:	

Read/Write Settings

This section contains value that can be passed to the lesson *and* received back from the lesson. These are the *Raw Score*, *Maximum Score*, *Lesson Status*, *Entry*, *Time*, *Attempt Number*, *Audio* setting, *Speed*, *Language*, *Lesson Type*, *Lesson Location*, and *Suspend Data*. *Lesson Status* and *Entry* have drop-down lists with available values.

As shown below, the values that have been *received* from the lesson are shown in green. You might then launch the lesson immediately or during the next session (after clicking the *Load Previous Values From Cookies* button) to see if the *Lesson Location*, *Suspend Data*, and other values were used transferred to and used appropriately by the lesson.

Read/Write Settings ^

Raw Score:	100
Maximum Score:	100
Lesson status:	completed
Entry:	resume
Time (HH:MM:SS):	00:02:8
Attempt Number (starts with 0):	0
Audio (1 to 100):	50
Speed (-100 to 100):	0
Language:	
Lesson Type:	
Lesson Location (Bookmark):	12
Suspend Data:	1_2_3_4_5_6_7_8_9_10_11_12

GetParam Messages

This section is populated once the lesson sends the *GetParam* message to AICC Watch. The return value from this message is displayed. The screen capture to the right corresponds with Read/Write settings returned in the previous session. For example, notice that the *Lesson_Location=12*.

GetParam Messages ^

```

error=0
error_text=Successful
aicc_data=
[Core]
Student_ID=44
Student_Name=Rhodes, Jeffrey
Lesson_Location=12
Credit=Credit
Lesson_Status=
Score=100,100,0
Time=00:02:8
Lesson_Mode=normal
[Core_Lesson]
1_2_3_4_5_6_7_8_9_10_11_12
[Student_Data]
Attempt_Number=0
Mastery_Score=80
Max_Time_Allowed=
Time_Limit_Action=continue,no message
[Student_Preferences]
Audio=50
Language=
Lesson_Type=
Speed=0
    
```

PutParam Messages

This section is populated when the lesson sends the *PutParam* message to AICC Watch. Each element *sent* by the lesson along with its value is shown in a table. The return value *from* AICC Watch is shown below the table. For example, in the capture below, the lesson sent back *lesson_location*, *lesson_status*, *Score* (AICC Watch splits it into min, max, and raw to correspond to the Read/Write Settings), *time*, *[core_lesson]*, and *audio*. It then returns the fact that the *error* was 0. Note that any values corresponding to the *Read/Write Settings* are updated in that section (and shown in green) as well.

PutParam Messages ^

Element	Value
lesson_location	12
lesson_status	completed,logout (Core.Exit)
ScoreMin	0
ScoreMax	100
ScoreRaw	100
time	00:02:8
[core_lesson]	1_2_3_4_5_6_7_8_9_10_11_12
audio	50

error=0
error_text=Successful

PutInteractions Messages

This section is populated when the lesson sends the *PutInteractions* message to AICC Watch. This is an important section as many issues with AICC communication relate to interactions. The various interaction values (Date, Time, Interaction ID, Objective ID, Interaction Type, Correct Response, Student Response, Result, Weighting, and Latency) are shown in a table. The return value *from* AICC Watch is shown below the table. An example is displayed below.

PutInteractions Messages ^

Date	Time	Interaction ID	Objective ID	Interaction Type	Correct Response	Student Response	Result	Weighting	Latency
2011/02/27	17:27:16	Faculty_2006_Flags	Faculty	matching	{1.u,2.u,3.a,4.d}	1.u,2.u,3.a,4.d	1	1	00:00:11
2011/02/27	17:27:22	Faculty_2007_Flash	Faculty	choice	{A,D,E}	{D,A,E}	1	1	00:00:04
2011/02/27	17:26:29	Faculty_1999	Faculty	choice	{A,B,C,E,F}	{A,B,C,E,F}	1	1	00:00:06
2011/02/27	17:26:40	Faculty_2001_Cheltenham	Faculty	choice	{A,C,D,F}	{A,C,D,F}	1	1	00:00:05
2011/02/27	17:27:01	Faculty_2004_NET	Faculty	choice	{A,C,D,E}	{A,C,D,E}	1	1	00:00:06
2011/02/27	17:26:50	Faculty_Books	Faculty	choice	{A,B,C}	{A,B,C}	1	1	00:00:07

error=0
error_text=Successful

PutObjectives Messages

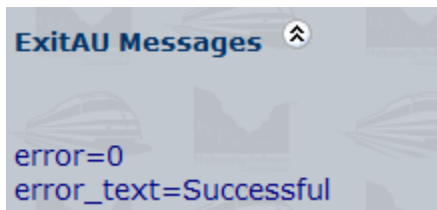
This section is populated when the lesson sends the *PutObjectives* message to AICC Watch. The data received from the lesson is once again displayed in a table. The return value *from* AICC Watch is shown below the table.

PutPerformance Messages

This section is populated when the lesson sends the *PutPerformance* message to AICC Watch. The data received from the lesson is once again displayed in a table. The return value *from* AICC Watch is shown below the table.

ExitAU Messages

This section is populated when the lesson sends the *ExitAU* message to AICC Watch. The return value *from* AICC Watch is displayed.



Most Recent Command

As the session progresses, the most recent AICC Command received is shown at the top of the screen. For example, in the screen capture below, the most recent command is *GetParam*.



Advanced Uses of AICC Watch

Architecture

The entire Visual Studio 2010 solution for AICC Watch is included with the product. This is for the internal use of the developer and is not to be resold or incorporated into other products. Let's look at each of the key files shown in *Solution Explorer* to the right.

AiccService.asmx. This is a web service called by JavaScript (Ajax) on the main (*default.aspx*) page. The service tells AICC Watch the last command received. When the command is new, the page reloads with the latest data.

AiccWatchModule.vb. This is for code that is used by *default.aspx*, *receivedata.aspx*, and/or *AiccService.asmx*.

Clientaccesspolicy.xml/crossdomain.xml. These files allow from cross domain calls from Flash and Silverlight respectively.

default.aspx. This is the main AICC Watch file. It is the one that you launch to start the program.

platte.js. This is a JavaScript file mainly used to show and hide the various sections when clicking the up and down arrow graphics.

receivedata.aspx. This is the target for the HTTP Post calls from the content. For example, the URL used to launch the lesson (AU), might look like this:

```
http://localhost/Tracker3Samples/TrainingStudio2/start.html?aicc_sid=a9cd1214-b031-40fa-b815-5fe26ee006d3&aicc_url=http%3a%2f%2flocalhost%2fAiccWatch%2freceivedata.aspx
```

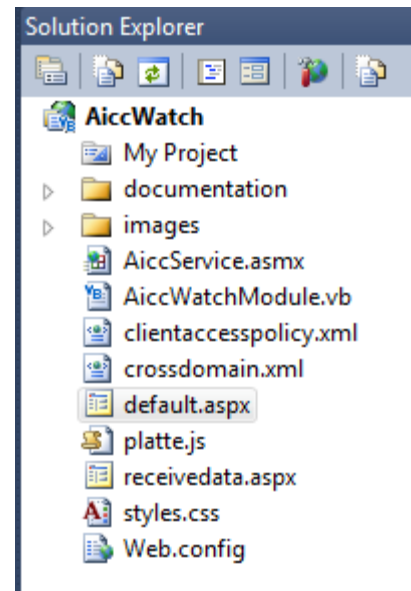
With URL encoding, it would look like this:

```
http://localhost/Tracker3Samples/TrainingStudio2/start.html?aicc_sid=a9cd1214-b031-40fa-b815-5fe26ee006d3&aicc_url=http://localhost/AiccWatch/receivedata.aspx
```

As we describe in the next section, you can take advantage of this to use Visual Studio to debug your AICC lessons.

styles.css. This Cascading Style Sheet (CSS) file has various styles for the look and feel of AICC Watch.

Web.config. This is a standard ASP.NET file. It does not have any user-configurable data in this case.



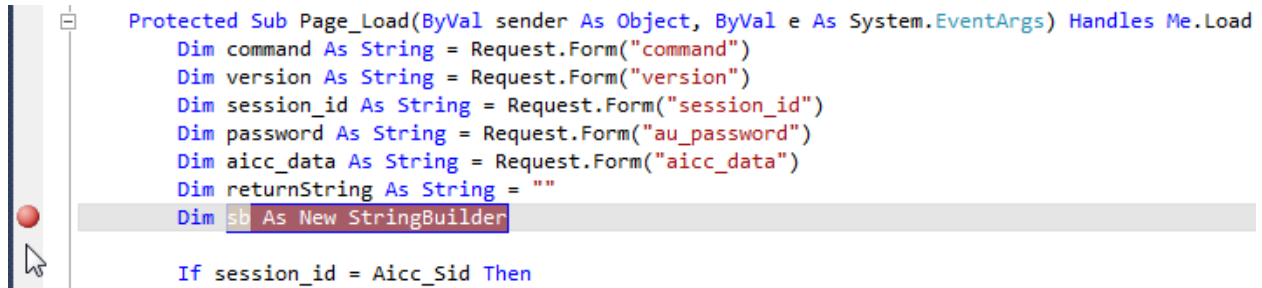
Running AICC Watch from Visual Studio

To run AICC Watch in Visual Studio 2010, double-click the *AiccWatch.sln*. You can then put in breakpoints to see the program in operation. For an example of doing this, see the next section.

Example: Debugging with Flash Builder and Visual Studio

We have found it useful to when troubleshooting AICC problems to use AICC Watch in debug mode. This allows us to put breakpoints in Flash Builder or Visual Studio for the content itself. Here is an example of doing this with Flash Builder.

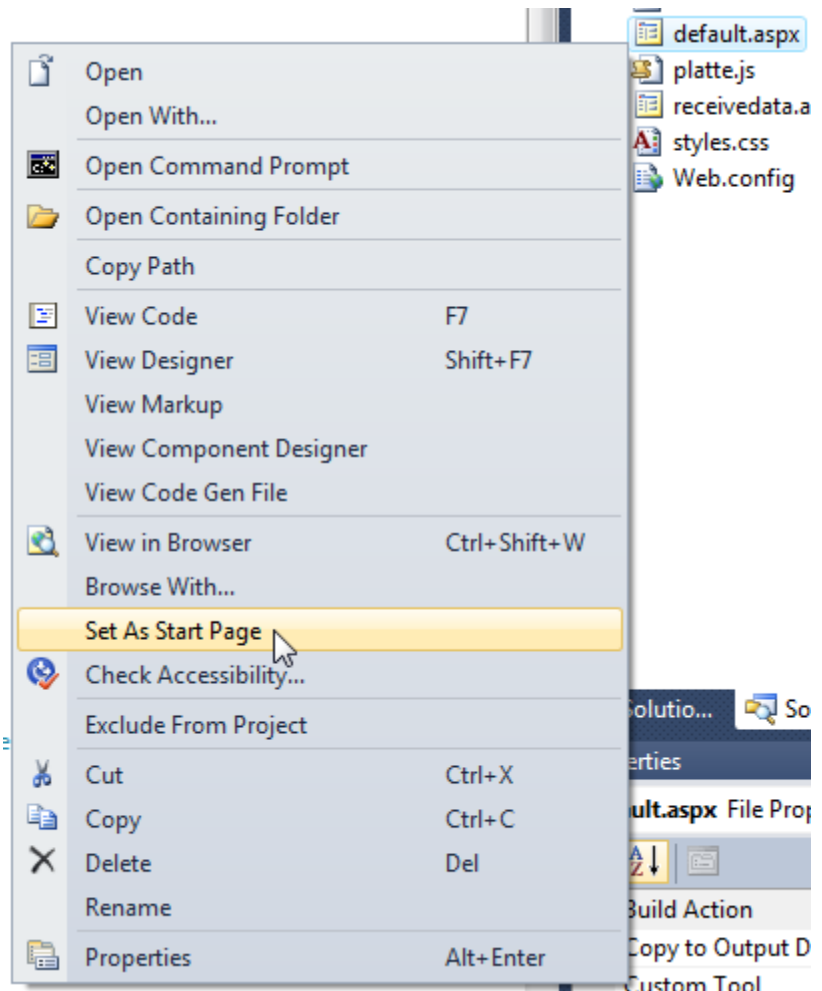
1. Put a breakpoint in *receivedata.aspx.vb* inside the *Page_Load* handler as shown below.



```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    Dim command As String = Request.Form("command")
    Dim version As String = Request.Form("version")
    Dim session_id As String = Request.Form("session_id")
    Dim password As String = Request.Form("au_password")
    Dim aicc_data As String = Request.Form("aicc_data")
    Dim returnUrl As String = ""
    Dim sb As New StringBuilder
    If session_id = Aicc_Sid Then
```

It is important that the breakpoint be before the *If session_id = Aicc_Sid Then* line. This is because we need to set the *Aicc_Sid* property to have the value passed in by the lesson before that line. Otherwise, the program will not continue.

2. Set *default.aspx* as the Start Page (see capture to the right) and launch it via Visual Studio in debug mode. It should already be the Start Page, but it doesn't hurt to set it again.
3. Go to your content development environment, in this case *Adobe Flash Builder*. We put a breakpoint in where there are issues. For example, in the screen capture below, we put a breakpoint in the function that is called when we receive the return value from the *GetParam* call. This is shown in the screen capture below.

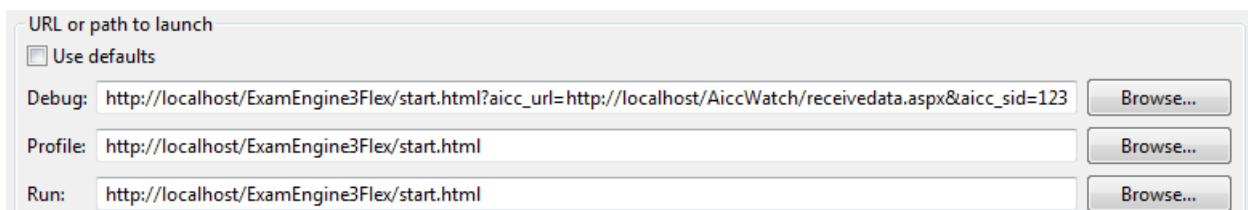


```

562         private function AiccComplete(e:Event):void {
563             var loaderId:URLLoader = e.target as URLLoader;
564
565             if (loaderId != null) {
566                 var xmlData:XML = new XML(loaderId.data);
567                 var cr:String = String.fromCharCode(13, 10);
568                 var dataArray:Array = xmlData.toString().split(cr);

```

- The issue comes that we need Flash Builder to initiate the session in order to debug it. So we launch it with an appropriate set of parameters and then use Visual Studio to both see the values sent from Flash Builder and to set the variables correctly so that it looks like a valid session. This is shown below.

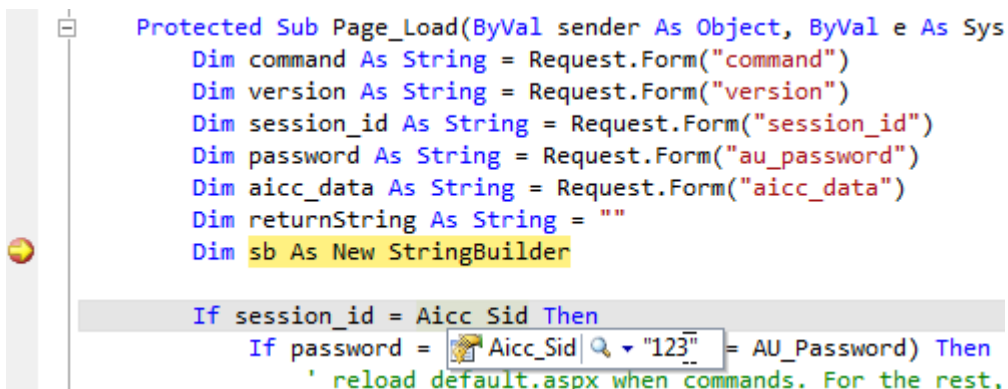


Here is the complete URL used:

`http://localhost/ExamEngine3Flex/start.html?aicc_url=http://localhost/AiccWatch/receivedata.aspx&aicc_sid=123`

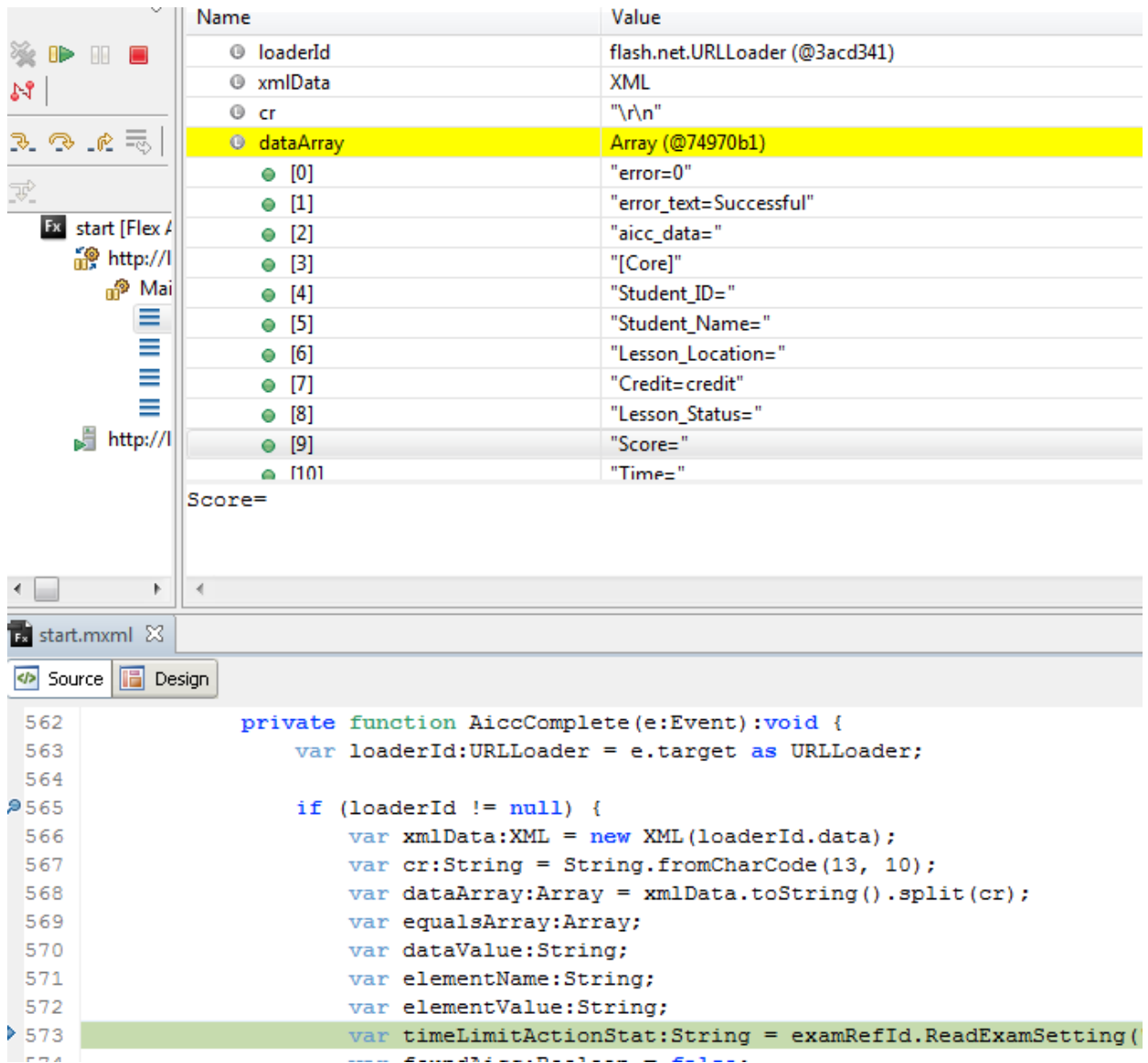
Notice how we point to `http://localhost/AiccWatch/receivedata.aspx` for the `aicc_url` and then just set an easy value to type (`123`) for the `aicc_sid`.

- Launch the lesson in Flash Builder. This will cause AICC Watch in Visual Studio to stop at its breakpoint (see step 1). We need to set the value of `Aicc_Sid` to `123` as shown below so that AICC Watch returns actual data and not an error message.



Then continue execution in Visual Studio.

- We can then debug the return value in Flash Builder as shown below.



Notice how we are able to see the return data from AICC Watch in split it into the *dataArray* variable shown above.

7. Finally, we can use Visual Studio to check inputs in the event of issues. For example, the screen capture below shows *PutInteractions* data sent to AICC Watch from Flash Builder.

```

Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    Dim command As String = Request.Form("command")
    Dim version As String = Request.Form("version")
    Dim session_id As String = Request.Form("session_id")
    Dim password As String = Request.Form("au_password")
    Dim aicc_data As String = Request.Form("aicc_data")
    Dim returnString As String = ""
    Dim sb As New StringBuilder

    If session_id = Aicc_Sid Then
        If password = "" OrElse (password = All_Password) Then

```

100 %

Locals

Name	Value
Me	{ASP.receivedata_aspx}
aicc_data	""date", "time", "interaction_id", "objective_id", "type_interaction", "correct_response", "student_response", "
command	"PutInteractions"
e	{System.EventArgs}
password	Nothing
returnString	""
sb	Nothing
sender	{ASP.receivedata_aspx}
session_id	"123"
version	"3.0"

Note that the information is not displayed in the normal AICC Watch fields in this scenario since AICC Watch did not actually launch the content. But through the combination of Visual Studio and Flash Builder debugging, we are able to get our AICC calls exactly right.