# Creating Adobe LiveCycle Projects in Xcelsius 2008

## Applies to:

Xcelsius 2008

## Summary

This document covers the basic steps needed to install the Adobe LiveCycle Data Services and Xcelsius LiveCycle Data Services (LCDS) Connector, use FlexBuilder and Eclipse to create a new LCDS project and some sample feeds, deploy the LCDS project to Tomcat, and finally, use the LCDS Connector to attach an Xcelsius model to the feed for real-time data stream visualization.

**Author(s):** Kiet Trang, Technical Account Manager

**Company:** SAP

**Created on:** 01 November 2008


© 2008 SAP AG

## Table of Contents

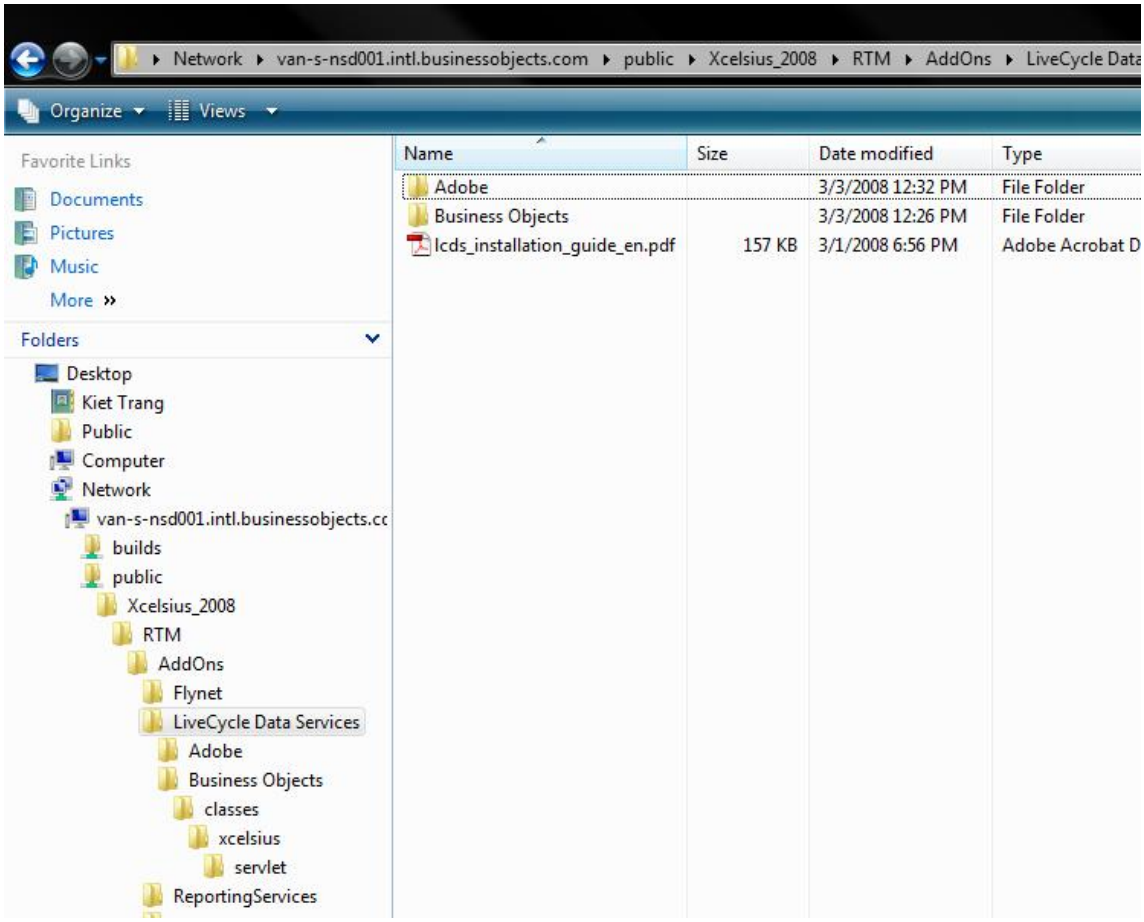
Install LCDS and Connector ..... 3
Create a LCDS Application with FB3 / Eclipse ..... 4
Sample Code ..... 8
Real-time Data Stream Visualization ..... 11
Attaching an Xcelsius model to the data feed ..... 11
Creating the Xcelsius Dashboard ..... 12
Related Content ..... 13
Xcelsius Add-on Marketplace Copyright ..... 13
Copyright ..... 14


© 2008 SAP AG

## Install LCDS and Connector

LiveCycle Data Services 251 and Connector ship with Xcelsius Engage Server and Business Objects Xcelsius Enterprise 2008. You can find them in the AddOns/LiveCycle Data Services folder of the Xcelsius media. The Adobe folder contains the LCDS 251 install exe file (Figure 1). Double-click the install.exe file and do not type a keycode when prompted. This will install the Express version which is fully functional but limited to one CPU, no clustering, and load balancing. Refer to the licence.txt file that is installed with LCDS 251 for more information.

**Figure 1**




The Business Objects folder contains the files that need to be installed and configured with LCDS to allow it to work with the Connector (that is, the servlet which is hosted by the LCDS application and provides the Connector with information on the available destinations or feeds that can be used with Xcelsius and the associated meta-data about the data being surfaced through a particular destination).

Once you install LCDS 251 you will see a screen similar to Figure 2.

The samples.war is a sample application which contains out of the box demos and sample code to help you get started. The flex-admin.war is an application that helps you manage andmonitor your LCDS applications. Flex.war is the skeleton war file – that is, it contains the bare essentials needed to develop a LCDS application. You would typically use this as a basis to develop any LCDS application that will be put into production. If you are just building a demo then you might want to use the samples.war as a starting point. LCDS is essentially a Java web application and everything is encapsulated in a war file.


© 2008 SAP AG

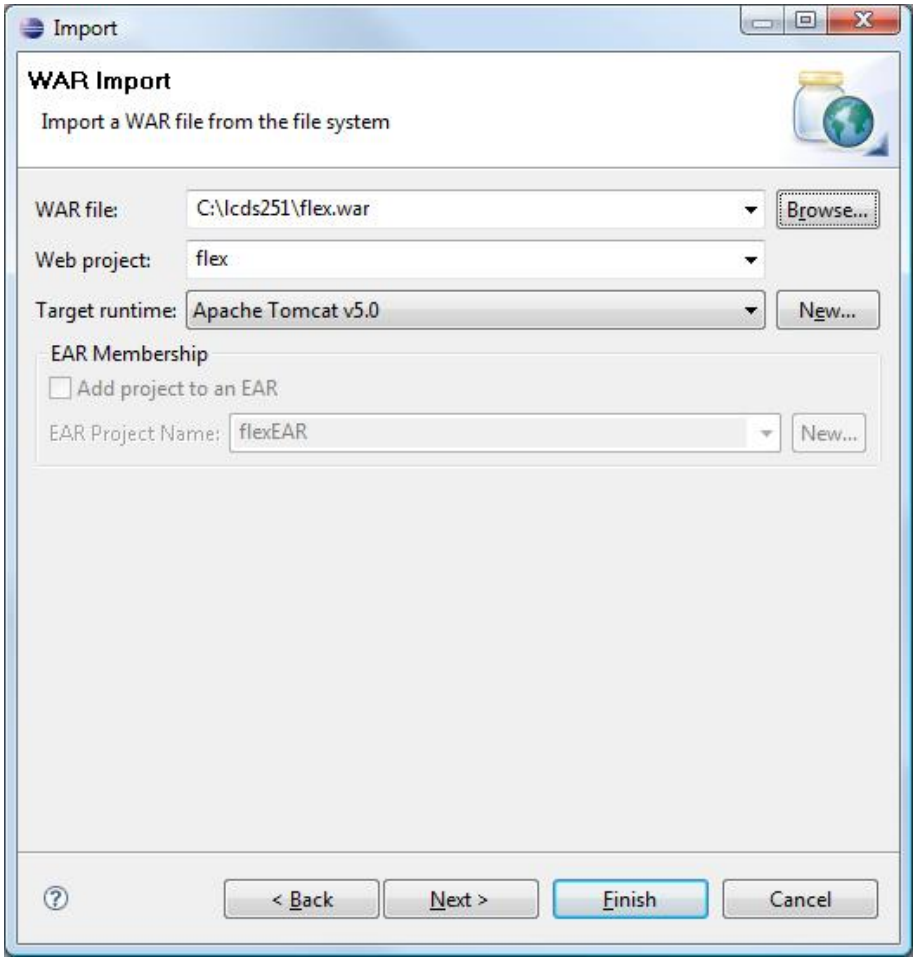
## Create a LCDS Application with FB3 / Eclipse

FB is an Eclipse-based IDE and is available as a complete install or as a plugin install that you can install on an existing Eclipse installation. It is recommended that you perform the plugin install on an Eclipse WTP 2.01 install in order to take advantage of the tooling in the WTP that helps you with web and Java development. To strictly work with LCDS you only need to have an Eclipse WTP install. FB3 is only needed if you want to work with Flex.

This document assumes that you are using a FB3 / WTP combination together with Tomcat. Complete these steps to create an LCDS from the skeleton flex.war file:

1. Ensure that you have set up the Tomcat server environment and have configured Eclipse to point to it (refer to the Eclipse documentation for more information).
2. Import the flex.war application into Eclipse to create a base Java web project (Figure 3).

**Figure 3**

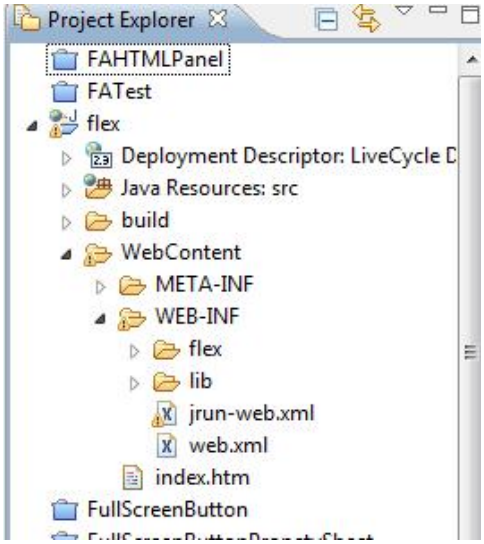



3. Click Finish (you do not need to import any of the web libraries on the next dialogue). You now have a Java web project with the base LCDS framework.
4. Import the server elements of LCDS Connector components:
5. Import the Xcelsius servlet classes and connector configuration files.
6. Merge the Xcelsius servlet declaration into the web.xml.
7. Import the xerces and xalan libraries.


© 2008 SAP AG

8. To import the Xcelsius servlet classes and configuration files, right-click on the "WEB-INF" folder of the web project and select Import > General > File System (Figure 4).
9. Browse to the Xcelsius media files and click the Business Objects folder.

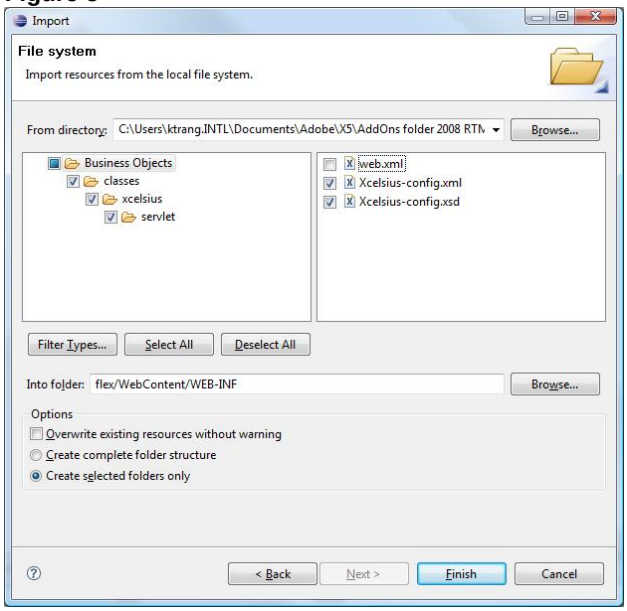
**Figure 4**




10. Select the classes folder and Xcelsius-config xml and xsd files (Figure 5) and then click **Finish**.

The classes folder contains the Xcelsius servlet that helps tell the Xcelsius designer which feeds are available to be used with Xcelsius and meta data regarding a particular feed. The Xcelsius-config.xml file allows you to configure which feeds are exposed to be consumed by Xcelsius and to define the meta-data for the feed (that is, fields returned and data types which will help the user Xcelsius designer/users understand the data being returned by the connector and appropriately map and bind these to the Xcelsius model). The Xcelsius-config.xsd file allows users to validate their Xcelsius-config.xml against, to make sure they have properly set it up.

**Figure 5**

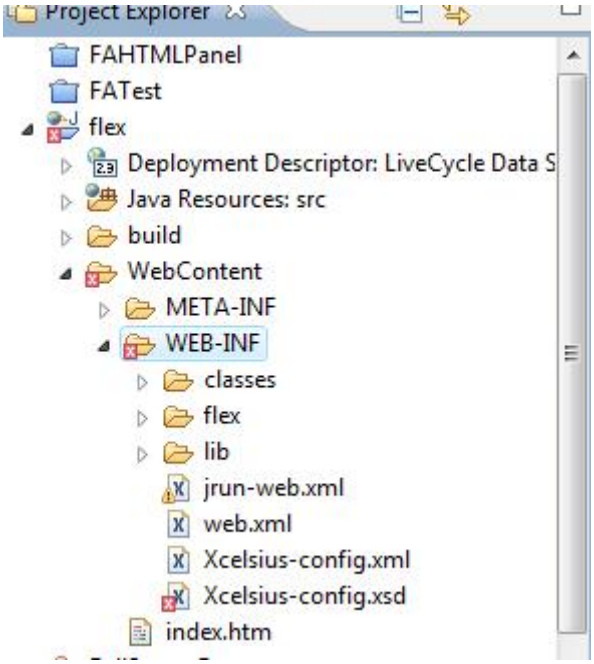




© 2008 SAP AG

After selecting the folder and files, the Project Explorer panel similar to Figure 6 appears.

**Figure 6**

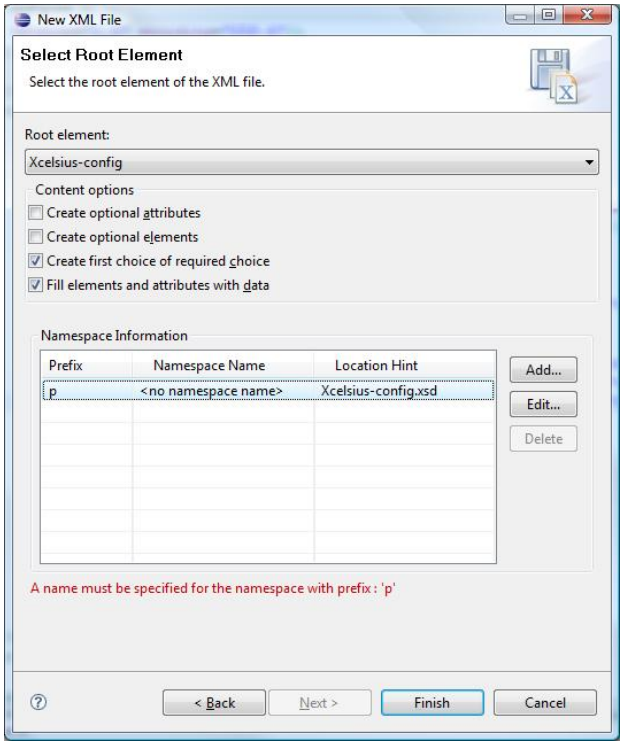



Note that the Xcelsius-config.xsd shows a red “x”. At the time of writing this document there are some errors in the XSD file and a bug has been submitted on this. To correct these errors, download the Xcelsius-config.xsd file in the Xcelsius 2008 LiveCycle Data Services supporting files package listed in the Related Content section of this document.

11. Replace the Xcelsius-config.xsd file that was imported into your project with this file. Optionally, you can also use Eclipse/WTP to to generate a blank/skeleton XML file from the XSD file by right-clicking the XSD file > Generate > XML file (Figure 7).
12. You will need to edit the default “prefix” and delete the “p”. Aside from validating the XML file for you, Eclipse will also give you code-completion which editing the XML file and prompt you with the possible properties and values.
13. Merge in the Xcelsius servlet declaration into the existing web.xml file inside the web project. The “Business Objects” folder from the Xcelsius install media has a sample web.xml with the required servlet declaration.


© 2008 SAP AG

**Figure 7**




Note at the time of writing this document, there is an error in the sample provided (this bug has been submitted). The correct servlet declaration is as follows:

```
<servlet>
     <servlet-name>XcelsiusServlet</servlet-name>
     <servlet-class>xcelsius.servlet.XLCDSServlet</servlet-class>
     <init-param>
         <param-name>configuration.file</param-name>
         <param-value>/WEB-INF/Xcelsius-config.xml</param-value>
    </init-param>
 </servlet>

 <servlet-mapping>
     <servlet-name>XcelsiusServlet</servlet-name>
     <url-pattern>/xcelsius/*</url-pattern>
 </servlet-mapping>
```

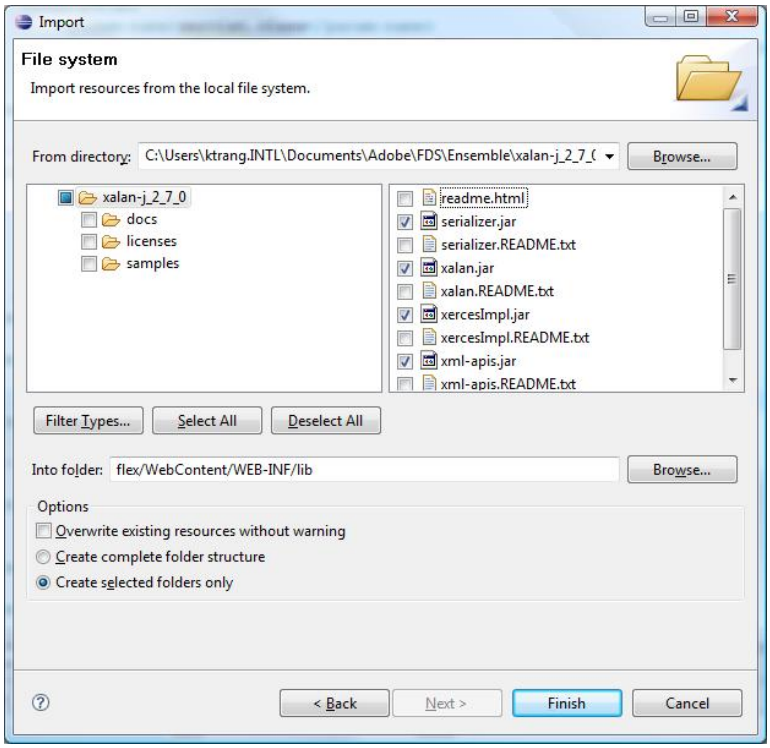
Refer to the web.xml file in the Xcelsius 2008 LiveCycle Data Services supporting files package listed in the Related Content section of this document. The sample web.xml file already has the servlet declaration inserted for your reference.

14. Since Xcelsius servlet leverages the standard xerces/xalan libraries to read the xml configuration files you will need to import these libraries into the project. You can download the library directly from Apache (http://xerces.apache.org/xerces-j/).


© 2008 SAP AG

15. Right-click the lib folder of the web project and go to import > General > File System and then browse to where you unzipped the xalan-j library and select the serializer.jar, xalan.jar, xercesImpl.jar, and xml-apis.jar files.

**Figure 8**




Now that you have imported all the Connector server elements into your web project, you are ready to configure LCDS and point it to data sources such as JMS queues or to start developing your own adaptors.

16. Refer to the LCDS documentation and guides for details, and reference materials on the API (http://blogs.adobe.com/flexdoc/2007/06/livecycle_data_services_docume.html).

## Sample Code

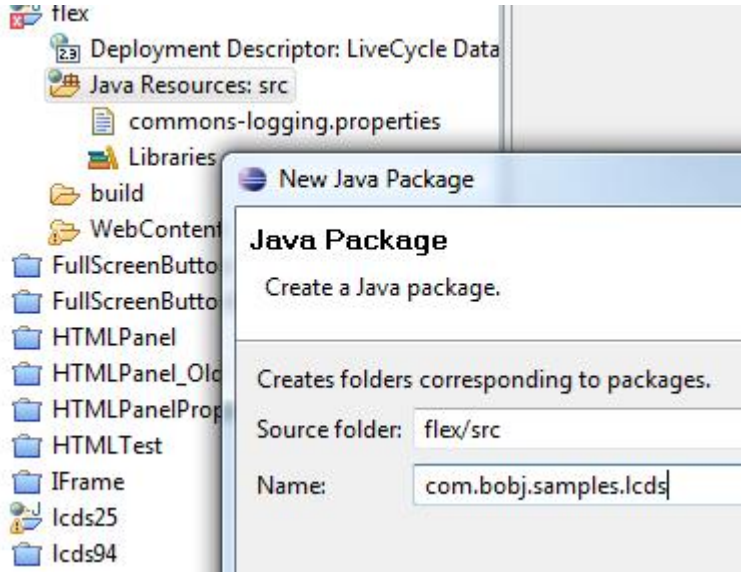
See the file RandomNumberFeed.zip in the Xcelsius 2008 LiveCycle Data Services supporting files package listed in the Related Content section of this document. The sample code that will help you get started. This sample shows you how to create a custom (data feed) service. The data service in this case connects to a simple class that generates random numbers and then drops the data payloads into the service's message queue which then is surfaced through LCDS as a data destination/feed.

To add the sample code to your project, complete these steps:

1. Unzip the file.
2. Right-click the web project's Java Resource folder and then go to New > Package and name it "com.bobj.samples.lcds" (Figure 9).


© 2008 SAP AG

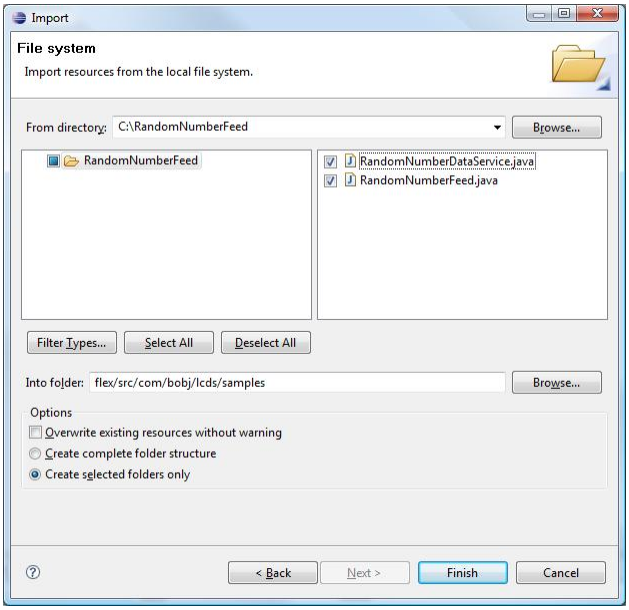
**Figure 9**




3. Right-click the com.bobj.samples.lcds package folder and go to import > General > File System.
4. Select the folder that contains the unzipped .java files and then select both Java classes (Figure 10).
5. Register the sample data service class with LCDS by opening up the services-config.xml file located in the “WEB-INF/flex” folder and add the following entry to declare the service:

```
<service
        class="com.bobj.samples.lcds.RandomNumberDataService"
        id="random-number-ds">
  </service>
```

**Figure 10**




6. Register the service by opening up the Xcelsius-config.xml file and adding the following entry

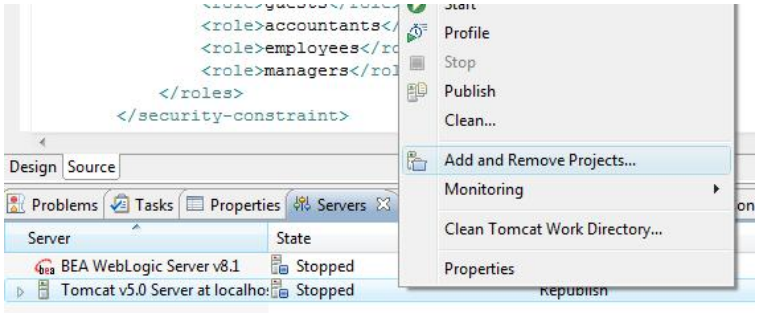

© 2008 SAP AG

```
<destination>
    <id>RandomNumbers</id>
    <description>Sample LCDS Feed</description>
    <metadata>
      <field datatype="String" shape="Singleton">time</field>
      <field datatype="Short[]" shape="1D">column/row</field>
      <field datatype="Short[]" shape="2D">table</field>
    </metadata>
  </destination>
```

The service is now visible to the Connector and meta-data is added to describe the data payload to the Connector. You are ready now to test the LCDS and the sample data feed

7. Make you have compiled the project in eclipse, then deploy it to the test Tomcat server environment inside of eclipse, and start Tomcat.
8. Right-click **Tomcat** in the Servers panel, and then click **Add and Remove Projects** (**Figure 11**).

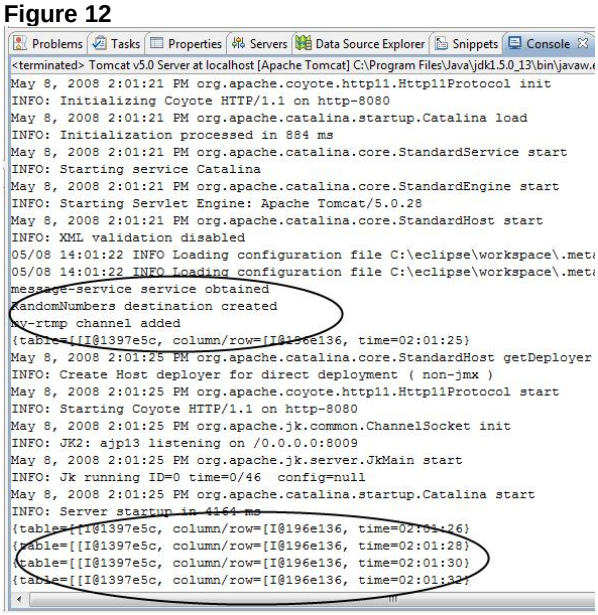
**Figure 11**




9. Add your LCDS web project to the server.
10. Right-click **Tomcat**, click **Publish** to make sure the test environment has the most updated copy of your project, and then click the green play button on the Servers panel to start up Tomcat.

If everything was set up correctly you see output similar to Figure 12.

**Figure 12**





© 2008 SAP AG

Note the debug message in the first circle there will tell you if the sample data service/feed was successfully created at the start up of the application. Messages below are outputted each time a data payload is dropped into LCDS. At this point you have confirmed that the LCDS and the sample data service are working correctly; however you need to confirm that the server end of the LCDS Connector is work as well.

11. In a browser window, go to your application (for example, http://localhost:8080/flex/xcelsius).

You should receive an XML document that describes the data feeds and destinations that have been made available to Xcelsius to consume:

```
<?xml version="1.0" encoding="UTF-8"?>
<destinations>
   <channel>
          <id>my-rtmp</id>
          <url>rtmp://localhost:2038</url>
          <type>mx.messaging.channels.RTMPChannel</type>
   </channel>
   <destination>
          <id>RandomNumbers</id>
          <description>Sample LCDS Feed</description>
          <metadata>
                 <field datatype="String" shape="Singleton">time</field>
                 <field datatype="Short[]" shape="1D">column/row</field>

                 <field datatype="Short[]" shape="2D">table</field>
          </metadata>
          <channel id="my-rtmp" />
   </destination>
</destinations>
```

This is the information that the Xcelsius designer will use to connect and bind to a particular data feed/destination. This information is not something an Xcelsius user needs to worry about, but is provided here for information purposes only. At this point you can export you web project to a .war file and deploy this to an external Java application server or simply continue to work with the Eclipse Tomcat test/dev environment. To export to a war, right-click the web project and click **Export.**

## Real-time Data Stream Visualization

Use the LCDS Connector to attach an Xcelsius model to the feed for real-time data stream visualization.

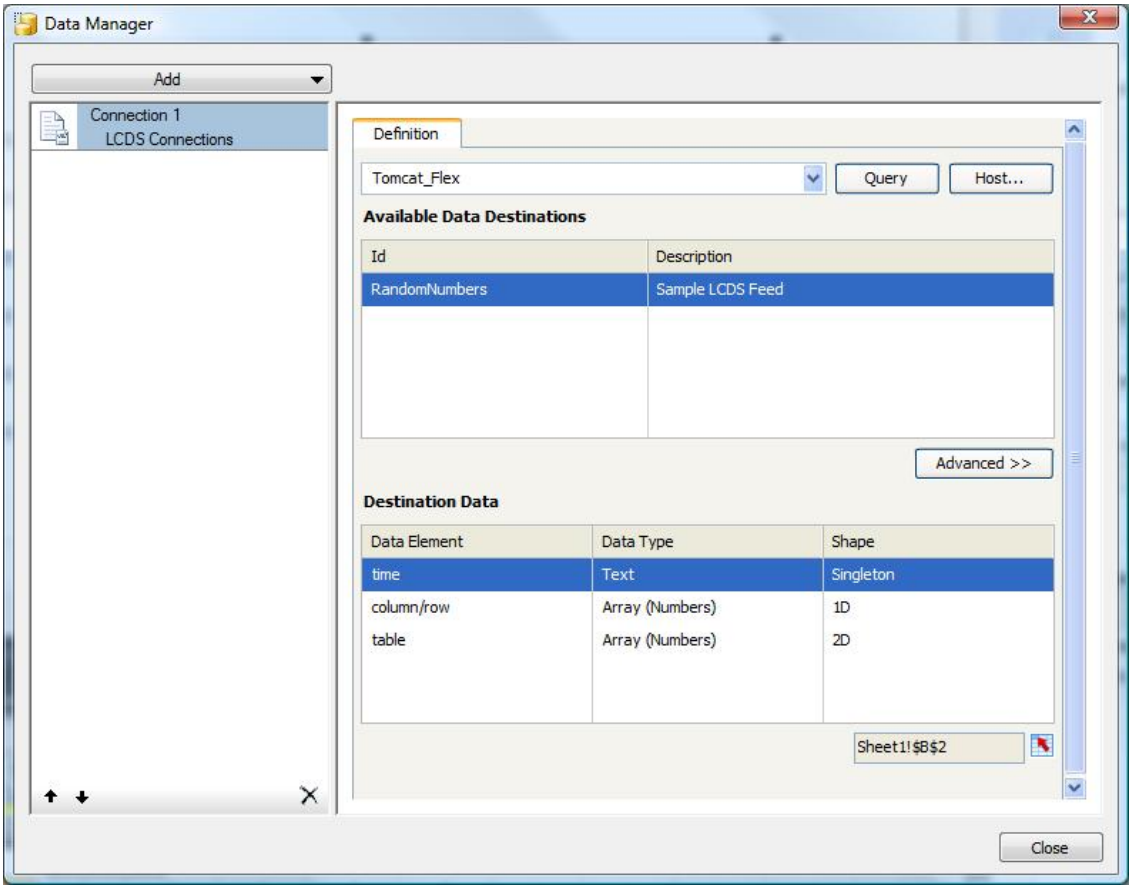
### Attaching an Xcelsius model to the data feed

1. Make sure that your sample LCDS application or web project is up and running and that you have the Flash Player version 9 or later installed.
2. Open the simple.xlf file in the Xcelsius 2008 LiveCycle Data Services supporting files package listed in the Related Content section of this document.
3. attached simple.xlf in Xcelsius.
4. From the **Data** menu go to **Connections** > **Add** > **New LCDS Connection.**
5. Click the **Hosts** button and type a name for your LCDS host (you can set up more then one LCDS host inside of Xcelsius) and then type the url to the Connector servlet (for example, http://localhost:8080/flex/xcelsius). The LCDS Connector servlet URL is the only thing that the Xcelsius user requires, much like a WSDL URL when working with web services. All the other steps described above in this document are steps that will be performed by the LCDS administrator and/or developer.


© 2008 SAP AG

6. Click **OK** until you see the dialog box in Figure 13. Once you click on the query button, Xcelsius will retrieve the XML which describes the feeds

**Figure 13**




7. Click **RandomNumber** to retrieve the available data elements for this particular feed.
8. Click the **time** data element which in this case is a Singleton or in other words is just a single value to B2 on the spreadsheet.
9. Click the **column/row** data element, which is a 1D array (a row of 10 values), and bind this to C5:L5.
10. Click the table element, which is a 2D array (a 10 x 10 table of values), and bind this to C19:L28,
11. Click the **Close** button.

## Creating the Xcelsius Dashboard

1. Select the Gauge, bind the Title to B2 (the time value) and the Data to C5 (the first data value of the "column/row" data element).
2. Click the **Preview** button to see the Gauge change value.
3. Select the Column Chart and bind the Chart Title to the B2 (the time value) and Data by Range to C5:L5 (the "column/row" data element).
4. Click the **Preview** button to see the data in your model dynamically change. You can also use the LCDS Connector with Xcelsius's History component to help show trending over time (please refer to the Xcelsius documentation for more details on the History component as it is not specific to LCDS)
5. Select the History 1 component, bind the data to B2 and Data Destination the C2:L2 (this is where Xcelsius will keep track of the older values).


© 2008 SAP AG

6. Select the History 2 component, bind the data to C5 and the Data Destination to C6:C15.
7. Select the Line Chart, bind the Data by Range to C6:C15, make sure **Data in Columns** is selected, choose the **By Series** option, and bind the Category Labels to C2:L2.
8. Click the **Preview** button to see the line chart get populated over time.
9. Select the Filled Radar Chart and then bind the **Data by Range** to C19:L28.
10. Click the **Preview** button to see the radar chart dynamically populated.

## Related Content

Xcelsius 2008 LiveCycle Data Services Connector Files

SAP BusinessObjects Xcelsius Community

Sample Xcelsius Dashboards

Xcelsius Add-on Marketplace


© 2008 SAP AG

## Copyright

© 2008 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, System i, System i5, System p, System p5, System x, System z, System z9, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, POWER5+, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.