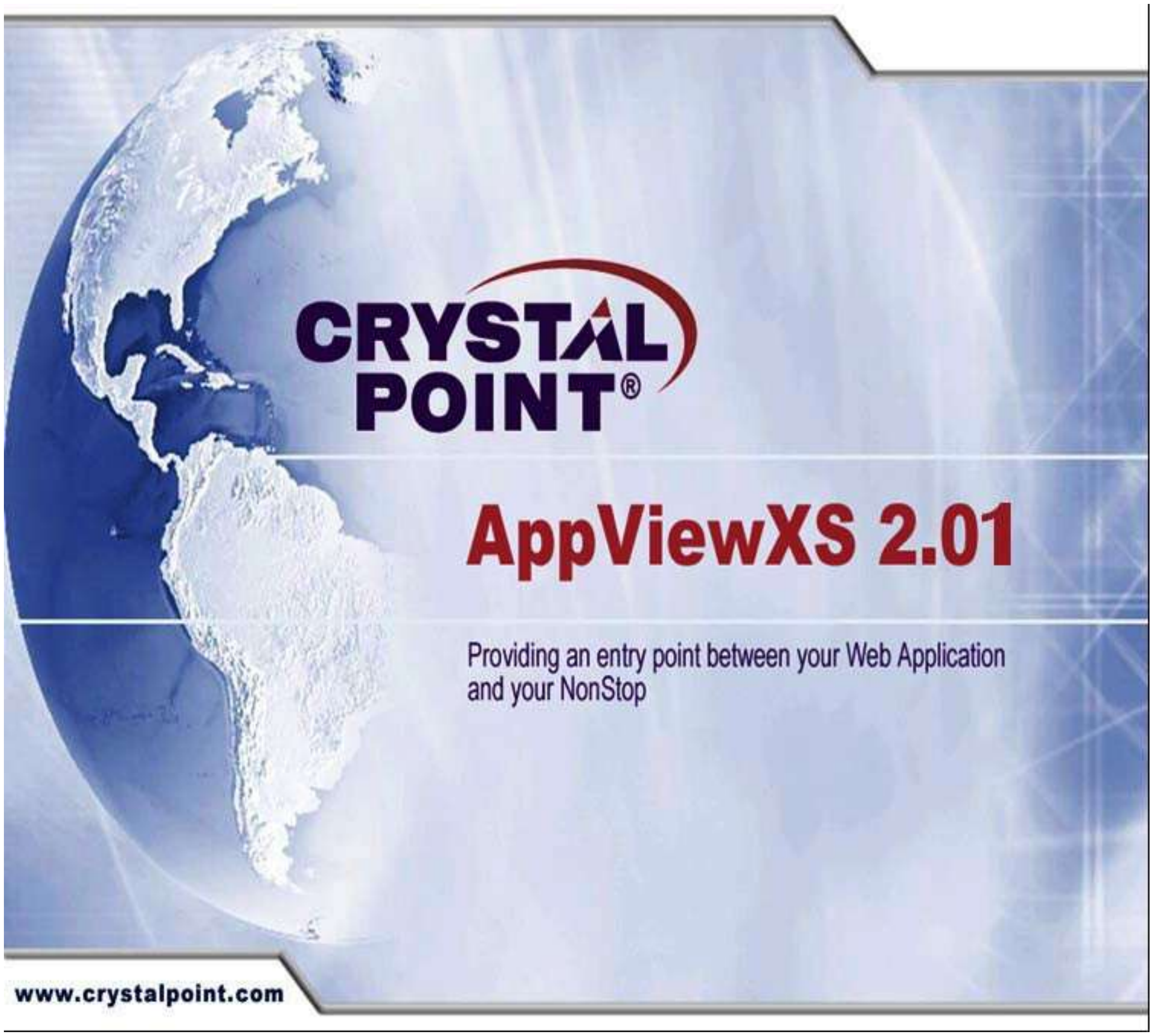


CRYSTAL POINT®

AppViewXS 2.01

Providing an entry point between your Web Application and your NonStop

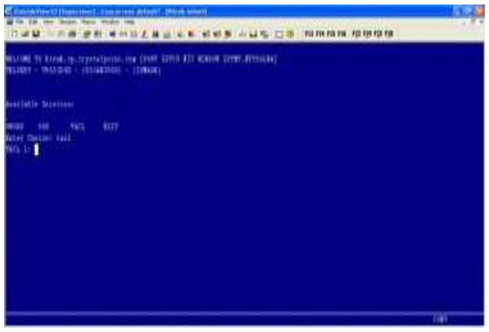
www.crystalpoint.com

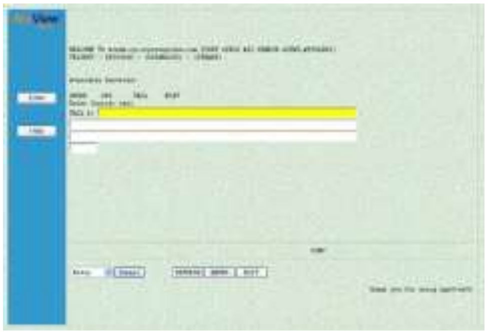
Table of Contents

AppViewXS Overview

AppViewXS, offered by Crystal Point, Inc., provides the capability for “green screen” legacy systems with all their advantages to be rapidly and securely published into the world of the Web via portals, while automatically* acquiring a modern, graphical user interface in the process.

Before: After





AppViewXS does this by using patented technology to transform 6530 data streams into a series of objects imbedded into an easily-modified HTML page serving as a global template. This transformation is done completely externally to the legacy application itself, and thus requires absolutely no modifications of any kind whatsoever to any portion of the host system or legacy application.

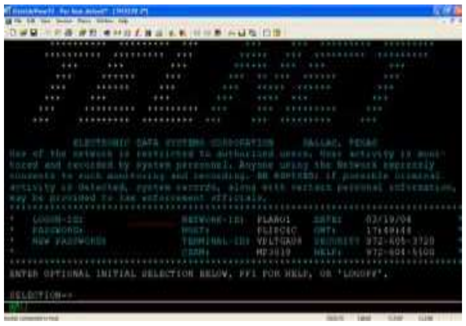
AppViewXS also provides a means to present users with a seamlessly blended environment combining legacy applications and java applications. This can greatly facilitate incremental Java development, whether you are migrating a legacy application to a web-tiered architecture, or merely adding java-based functionality as an enhancement to the legacy application.

AppViewXS is a server based, thin-client product. That means no jvm at the end user platform and no applets being downloaded to the end-user machine. Users are completely platform independent, requiring only a browser.

*In addition to automatic, transformation from green-screen to GUI, AppViewXS provides tools for advanced interface re-engineering of legacy applications. By “interface re-engineering” we mean:

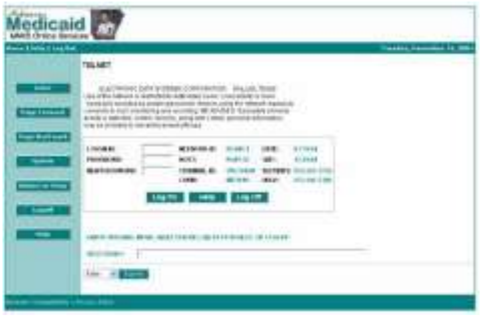
Accessing functionality embedded within terminal-based applications through graphical user interfaces characterized by modern controls, objects and communication protocols to allow a rejuvenated look and feel for the application as well as the ability to extend and build upon the legacy application for workflow and interface optimization as well as multi-hosted, multi-application data integration.

Before: After:



ENTER OPTIONAL INITIAL SELECTION BELOW, PF1 FOR HELP, OR 'LOGOFF'.

SELECTION==>



AppViewXS Integration Layer Installation Overview

The ability to integrate legacy applications and java applications is not required by all users of AppViewXS. Should the using organization 'merely' wish to rapidly and securely publish a legacy application onto the web, providing it with a modern Web look-and-feel, the Integration layer (Java Struts framework) is not needed.

Only when the using organization wants to present a blended combination of legacy application screens (fueled by the legacy application) and java screens (fueled by java application logic) does the Integration Layer become desirable. This blended presentation might be needed when creating new java logic to enhance the legacy application, or when migrating a legacy application from a single-tiered host application to a multi-tiered (also referred to as web-tiered) application. (e.g., the application database remains NonStop SQL, while the application logic is rewritten in the OSS iTP or other Java environment).

Installation of Integration layer: AppViewXS Struts Folder

AppViewXS does not contain or require a proprietary web server or web application server. Rather, AppViewXS was built using standard java architecture to permit installation into the widest possible range of web infrastructures including, almost certainly, yours. If, after reading these instructions, you have any difficulty or any questions about installing AppViewXS, you are urged to contact our technical support department, at support@crystalpoint.com or 425.806.1143.

The installation CD contains a folder named: appviewxs_struts. This folder is a self contained build environment that organizations can use with ANT to build the AVXSIntegrationAppWeb.war and AVXSIntegrationApp.ear files. Organizations may also modify the provided build.xml file to suit their specific needs for building a custom AVXSIntegrationAppWeb.war file.

The structure layout of the appviewxs_struts folder consists of:

- **build** folder – this folder contains the output for the build process.
- **documentation** folder – this folder contains the integration instruction guide (this document).
- **lib** folder – the lib folder contains javax.servlet.jar, commonst logging.jar, and struts.jar files that are needed to compile the source code.
- **src** folder – this folder contains the integration source code for AppViewXS using Struts. This is where your source code will need to be placed if you add your own code. During the build process, the output of the class files will be placed in the web/WEB-INF/classes folder.
- **web** folder – this folder contains the AppViewXS web application and gets copied over into the build folder during the build process.
- **build.xml** file – this file contains the ANT build script to compile and package the AppViewXS Web Application.

Installation of AppViewXS 2.01 into Web Application Server

In the 1.0 version of AppViewXS, the installation was solely a Java InstallShield process, and the installed product did not conform to the J2EE Web Application Archive standard. However, the Java InstallShield process provided a greater flexibility for installing the product in various Application Servers and Environments. To preserve that flexibility, AppViewXS 2.01 has retained its Java InstallShield installation capability.

To provide an easier and more standardized installation, AppViewXS 2.01 now also contains an AVXSIntegrationAppWeb.war file that conforms to the J2EE Web Application Archive standard and can be used to install onto the selected web application server. Organizations just copy the AVXSIntegrationAppWeb.war file into the deploy folder of their application server. Then launch a browser and access:

http://[server]/AVXSIntegrationAppWeb/index.html

With the war file, organizations can deploy AppViewXS 2.01 onto different application servers with minimal effort. If the provided war file does not exactly suit your needs, you can create your own WAR package and make modifications to the provided source code to facilitate an installation package that is tailored for your custom needs. You can even change the name of the AVXSIntegrationAppWeb.war file by renaming to something else that suits your need and then deploy it onto your web application server.

The AppViewXS 2.01 installation process using the AVXSIntegrationAppWeb.war file is quite simple:

1) Copy and Paste the AVXSIntegrationAppWeb.war file into your Application Server's deployment directory. For example, on Tomcat, that folder would be "webapps".

2) Open a browser window and point it to: "http://[ServerName]/AVXSIntegrationAppWeb" and configure the AppViewXS installation by clicking on the link "AppViewXS Config Screen".

3) Enter the Web Application directory path location where the AVXSIntegrationAppWeb.war file is installed. The location of this directory is based upon where the Application Server extracted the AVXSIntegrationAppWeb.war file to create a folder named "AVXSIntegrationAppWeb. Part of the installation for AppViewXS requires the physical location of this expanded war file because the installation process performs a recursive search and replace of different variables to reflect your environment settings. The search and replace process can only be done once. So if you run into some problems, then you will have to delete the expanded AVXSIntegrationAppWeb folder and redeploy the AVXSIntegrationAppWeb.war file.

4) Enter the URL address that is used to access the AppViewXS servlet. The URL value will be the same as that used in step number 2 above. The URL address consists of the server address hosting the application server and then the web application name. The web application name is usually based on what the name of your war file is except without the .war extension.

5) After the Web Application directory path and URL address has been entered into the input field, the next installation step is the global search and replace logic. The global search and replace occurs automatically after you submit to the server the web application directory path and the URL address. The search and replace logic is configured by default to use an external file named "extensions.txt" which contain the names of the file extensions the search and replace logic will filter through to perform the search and replace function. The file "extensions.txt" is stored in the

WEB-INF folder. Inside this file you will see a list of the extensions that the current logic will look through and perform the search and replace action. You can comment out an extension by placing a ‘ # ‘ character in front of the extension name, for example (#html). The search and replace logic is configured to use the extensions.txt file but this behavior can be turned off. To turn this feature off, you will need to modify the web.xml file and change the param value to ‘false’ for the parameter ‘use_extension_file’. After the global search and replace logic is finished, you will need to copy the registration files (resqportal.reg and portaladm.reg) for AppViewXS into your web application:

> location_of_expanded_war_file_on_app_server/AVXSIntegrationAppWeb/WEB-INF/appviewxs folder.

6) Finally, you will need to configure the Servlets for AppViewXS in the web.xml file The included web.xml file already comes pre-populated with these servlets, but you will need to update the parameters to match your environment. AppViewXS consists of three different servlets. These servlets are:

- appviewxs servlet “com.attinc.resqsrv.ResQServer” – this servlet configuration is required and is used to run the terminal emulators on the server.
- monitor servlet “com.attinc.resqsrv.RQSMonitor” – this servlet configuration is optional and is used to monitor AppViewXS emulation sessions.
- studio servlet “com.attinc.resqsrv.PortalStudio” – this servlet configuration is optional and is used to customize maps for AppViewXS emulation session.

a) The configuration parameters for appviewxs servlet consist of 4 initialization parameters:

- **rqs_servlet_dir=C:\ApacheWebserver\jakarta-tomcat-5.0.25\webapps\AVXSIntegrationAppWeb\WEB-INF\appviewxs**
  This parameter is used to specify the absolute path to the AppViewXS servlets directory. **Note**: you will need to point this parameter to reflect your own specific setting of where you installed AppViewXS.
- **rqs_debug=0**
  This is AppViewXS debug parameter. Normally, it should be set to **0** (disabled). To enable this parameter, set it to **1**.
- **rqs_log_days=50**
  The number of days to keep log files on the server before AppViewXS deletes them. The default value is **50**. This parameter is not necessary for AppViewXS to run.
- **rqs_check_sessions=300**
  This parameter indicates how often AppViewXS should check (in seconds) for timed out sessions. Any sessions that are discovered as timed out will have their user seat returned to AppViewXS's connection pool. The default value for this parameter is **300**. This parameter does not need to be set in order for AppViewXS to run.

b) The configuration parameters for the monitor servlet consist of 3 initialization parameters:

- **web_dir**
  Absolute path to servlet directory where the AppViewXS servlet resides
  e.g." C:\ApacheWebserver\jakarta-tomcat-5.0.25\webapps\AVXSIntegrationAppWeb\WEB-INF\appviewxs"

- **app_url**
  URL to AppViewXS's Public Internet Directory
  e.g. "http://yourserver/AVXSIntegrationAppWeb"

- **time_out**
  The length of one Monitor session (in minutes). Default value is **30** minutes.

c) The configuration parameters for the studio servlet consists of 4 initialization parameters:

- **rqs_debug=1**
  This is the Studio debug parameter. Normally, it should be set to **1** (enabled). To disable this parameter, set it to **0**.

- **rqs_image_dir=absolute path to images directory**
  Specifies the location for the Studio to store images. The location is generally under the appviewxs/images folder of the Public Internet Directory.
  e.g. "C:\ApacheWebserver\jakarta-tomcat-5.0.25\webapps\AVXSIntegrationAppWeb\web\images"

- **rqs_starthtml_dir=absolute path to AppViewXS Public Internet Directory**
  Specifies the location of the Studio's com directory. The location is generally under the AppViewXS Public Internet Directory
  e.g. "C:\ApacheWebserver\jakarta-tomcat-5.0.25\webapps\AVXSIntegrationAppWeb\web"

- **rqs_codebase_url=URL to AppViewXS Public Internet Directory**
  Specifies the URL to the Studio's classes.
  The location is generally "http://yourserver/AVXSIntegrationAppWeb"

7) After the servlets have been configured, then you will need to stop and start your web application server so that these new values will take effect.

AVXSIntegrationAppWeb application archive structure

The AVXSIntegrationAppWeb archive file when expanded has a simple structure. The layout consists of the main AVXSIntegrationAppWeb folder. Inside this main folder, are the subfolders:

- **config** – this folder contains web pages with instructions to configure the AppViewXS application.
- **include** – this folder contains include files that are used to simplify development of jsp and html files by allowing inclusion of external files into jsp/html files. This allows easy update of the navigation bar, or of specific information on any page that includes the include directive statement.
- **script** – this folder contains the common.js file used to modify or enhance the functionality of the jsp file displayed in the browser. The common.js file is defined in the title.inc file.
- **theme** – this folder contains the theme and layout for the overlaying look and feel interface applied to the html data returned from the AppViewXS application.
- **web** – this folder contains AppViewXS specific client side applications (Studio and Admin applet), AppViewXS documentation, and html files to launch AppViewXS applications.
- **WEB-INF** – this folder contains the heart of AppViewXS 2.01 product. There are 4 subfolders in this folder in addition to the struts-specific files, and the web.xml file used for the AppViewXS 2.01 installation process.
- **WEB-INF\appviewxs** – this folder contains the AppViewXS application's server side components needed to run and customize the AppViewXS application. Also, the appviewxs folder is where licenses (portaladm.reg and resqportal.reg) for AppViewXS are placed.
- **WEB-INF\classes** – this folder contains the compiled classes that are used to create a "proxy" interface between the AppViewXS application and other (java) application. The Jakarta commons logging properties file "log4j.properties" is also situated in this classes folder.
- **WEB-INF\lib** – this folder contains the jar files that are needed to use the Struts Framework, taglibs for the jsp, and the appviewxs.jar file needed to run AppViewXS 2.01 in the integrated development environment.
- **WEB-INF\src** – this folder contains the avxsintegration source code package that can be expanded to enhance the existing AppViewXS 2.01 integration development so that it will function with a specific J2EE application. The source code in this folder can be used as an example to develop custom solutions using AppViewXS 2.01 as a part of the overall product solution.

Integrating AppViewXS 2.01 into a Single J2EE Web Application

Integrating AppViewXS 2.01 into an existing J2EE Web Application is relatively simple. As an example, a sample web application named **servlets-examples** will be used to integrate the AppViewXS web application into. The **servlets-examples** web application is a demo application that comes installed with the Tomcat 5.x or higher installation. Create a folder named appviewxs_temp on the web application server computer. Copy the

AVXSIntegrationAppWeb.war into this folder. Then using the jar utility, expand the AVXSIntegrationAppWeb.war file. The command to extract the war file is:

```
jar –xvf AVXSIntagrationAppWeb.war
```

The appviewxs_temp folder should have the following contents:

config folder

include folder

script folder

theme folder

web folder

WEB-INF folder

AppViewXS.jsp file

ConfigResultScreen.jsp file

ConfigScreen.jsp file

EntryScreen.jsp file

HostInfo.jsp file

index.html file

launch_appviewxs.html file

Copy the extracted files and folders listed above into the pre-existing web application. For this example, the files and folders should be copied into the **servlets-examples** web application. It is important to preserve the above folder structure hierarchy when copying the contents into the web application folder.

The web application **servlets-examples** contain the following contents:

images folder

WEB-INF folder

cookies.html file

reqparams.html file

sessions.html file

helloworld.html file

index.html file

reqheaders.html file

reqinfo.html file

The WEB-INF folder contains the following contents:

classes folder

web.xml file

1) Looking at the contents of **servlets-examples**, you need to re-name the existing index.html file or merge the contents of the index.html file with AVXSIntegrationAppWeb's index.html file.

2) Copy the folders and files for config, include, script, theme, web, AppViewXS.jsp, ConfigResultScreen.jsp, ConfigScreen.jsp, EntryScreen.jsp, HostInfo.jsp, index.html, launch_appviewxs.html into **servlets-examples** folder.

3) Looking at the WEB-INF folder, you need to either rename your existing web.xml file or incorporate the contents of the two web.xml files into one web.xml file.

4) Copy the contents of the WEB-INF folder over into the **servlets-examples**' WEB-INF folder. You will need to take over everything from the AVXSIntegrationAppWeb's WEB-INF folder into the **servlets-examples**' WEB-INF folder. Note that the src folder is only necessary if you decide to use the web application in your Development IDE.

The new structure of **servlets-examples** folder after copying over the AVXSIntegrationAppWeb's contents:

images folder

config folder

include folder

script folder

theme folder

web folder

WEB-INF folder

cookies.html file

reqparams.html file

sessions.html file

helloworld.html file

index_old.html file

reqheaders.html file

reqinfo.html file

AppViewXS.jsp file

ConfigResultScreen.jsp file

ConfigScreen.jsp file

EntryScreen.jsp file

HostInfo.jsp file

index.html file

launch_appviewxs.html file

The WEB-INF folder for the **servlets-examples** folder will have the following contents:

appviewxs folder

classes folder

lib folder

src folder

extensions.txt file

struts-bean.tld file

struts-config.xml file

struts-html.tld file

struts-logic.tld file

struts-nested.tld file

struts-template.tld file

struts-tiles.tld file

web.xml file

Once all the necessary files have been integrated into the **servlets-examples** web application folder, the **servlets-examples** can run the AppViewXS application or it can be re-packaged into a war file and deployed to other web application servers. To launch the application, you will need to open a browser to: "http://yourserver/servlets-examples". Remember that you will still need to finish running the installation process before you can actually start using AppViewXS.

Developer's Guide to using AppViewXS 2.01

The Struts Framework integrated with AppViewXS 2.01 provides a useful plugin point to integrate an AppViewXS-provided Legacy application into your overall product solution. This solution provides many benefits and enhancements to any projects that require Host Access to legacy applications.

Configuring Development IDE

A simple and straightforward method of using your Java IDE to develop your custom AppViewXS Struts application is to take the AVXSIntegrationAppWeb.war file and extract the contents from it using the jar –xvf command. Create a folder named AVXSIntegrationAppWeb and copy the extracted contents into this newly created folder. You can now configure your Java IDE to use this folder for your development. Don't forget to configure your web.xml, and to copy over the AppViewXS licenses (portaladm.reg and resqportal.reg) into the AVXSIntegrationAppWeb/WEB-INF/appviewxs folder.

AVXSIntegrationAppWeb's Global Search and Replace Logic

The AppViewXS application itself uses three variables as place holders throughout the application which needs to be replaced with the user's specific settings. These three variables in the application are $SERVLET_NAME, $URL_TO_SERVLET, and $URL_TO_APP. During the installation process for the AVXSIntegrationWeb module, these variables are replaced with user specified values.

$SERVLET_NAME

This parameter is the name of the servlet that launches the emulation session. In the AVXSIntegrationAppWeb installation, the servlet name has a default value of "appviewxs". Note if you change the value for servlet name, you will also need to make a change in the web.xml file entry for the servlet-name parameter for servlet "com.attinc.resqsrv.ResQServer".

$URL_TO_SERVLET

This parameter is the URL that will be used to point to the web application that contains the AppViewXS servlets. Note that this parameter value should not include the servlet name.

$URL_TO_APP

This parameter is the URL to the web application and sub-directory that contains AppViewXS's applets.

CPSearchAndReplace Tool

The AVXSIntegrationAppWeb.war module encases logic that is used to install the AppViewXS application. It has its own global search and replace tool (CPSearchAndReplace) that recursively searches for the variable place holders and replaces them with the user specified values. The code for CPSearchAndReplace tool is located in the avxsintegration.utility package. To keep the installation process simple, the installation process only asks the user the physical location of the web application directory path and the URL address to access the AppViewXS servlet.

The physical location of the web application directory path gives the CPSearchAndReplace tool a starting point to begin the search. The URL address to access the AppViewXS servlet value provides enough information to obtain the values for $URL_TO_APP and $URL_TO_SERVLET replacements. As stated previously, the $SERVLET_NAME has been pre-defined to be "appviewxs". With the replacement values for these three place holder variables, the CPSearchAndReplace tool starts its search from the web application directory path and recursively searches and replaces the place holder variables with the specified values.

The CPSearchAndReplace has three configuration parameter settings that can be used to perform case sensitive searches, turn on/off recursive search, and to limit the scope of file extensions to search. The case sensitive search and recursive search parameters are boolean variables, the file extension is a string array that contains the specific file types that will be searched. The file extension string array is configured by default to contain the extensions for .txt, .html, .htm, .ini, .properties, .jsp, and .xml. These are the file types that contain the place holder variables in the AppViewXS application. Another feature of the CPSearchAndReplace tool is that you can also read in the file types by using a file named “extensions.txt”. The “extensions.txt” file is located in the WEB-INF folder. In this file you will need to enter the file types without the asterisks in front of the .extension and after each extension entered; a carriage return is used to separate the next extension. For example:

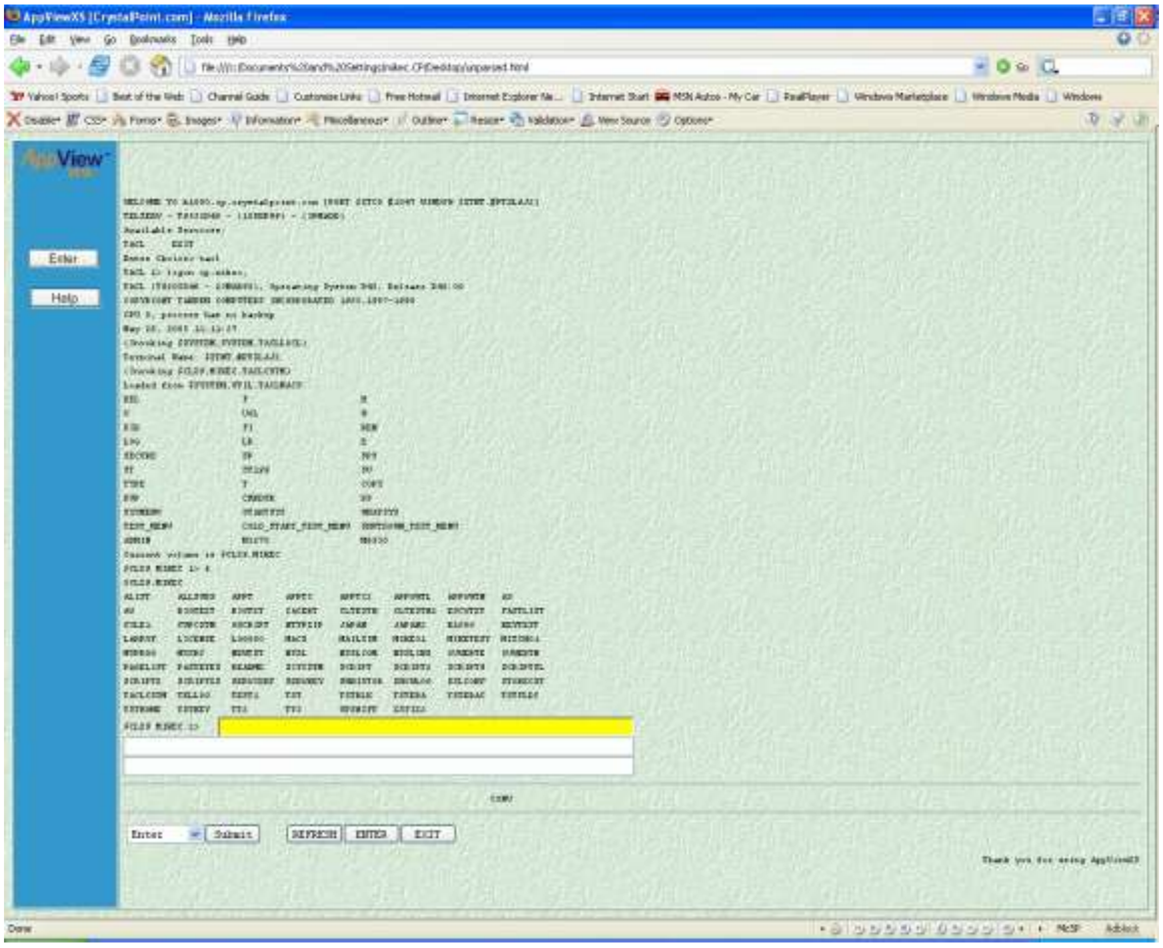
```
.txt
.html
.xml
```

You can also comment out an extension by using a # character (#.txt). By default, the CPSearchAndReplace tool is configured to read the file extensions from the extensions.txt file. This feature is configured in the web.xml file in the Action servlet configuration section. In the web.xml file, you should see a param setting for “use_extension_file”. This is where you can toggle the feature to use the external extensions file.

The place holder variables are specific for the AppViewXS application and the location of the web application directory provides a convenient starting point to begin the search operation. However, if you integrated the AppViewXS 2.01 product into your own existing web application and you do not want to perform a recursive search through your own application, then you will need to modify the code (avxsintegration.actions.AVXSConfigScreenAction) to search through only the AppViewXS application specific folders. If you are looking at an expanded view of AVXSIntegrationAppWeb module, the folders of interest would be the **web** folder and the **appviewxs** folder inside of WEB-INF.

HTML Content Returned from AppViewXS

The AppViewXS Struts logic takes the HTML content returned from an AppViewXS emulation session and parses it for specific contents. An example of an HTML content returned from an AppViewXS session will look like this:



The HTML source code:

```
<HTML>
<HEAD>
    <!----------BEGIN-META TAGS---------->
    <META NAME="GENERATOR" Content="AppViewXS">
    <META name="copyright" Content="Copyright © 2002 CrystalPoint.com, Inc.">
    <META http-equiv="Content-Type" Content="text/html; charset=iso-8859-1">
    <META http-equiv="Content-Language" Content="en-us">
    <META http-equiv="Cache-Control" Content="no-cache">
    <!----------END-META TAGS---------->
    <TITLE>AppViewXS [CrystalPoint.com]</TITLE>

<SCRIPT LANGUAGE="javascript" Type="text/javascript">
<!--
```

```
function customSubmit(sKey) {
 document.forms['RQSDefaultForm'].rqs_functionID.value = sKey;
 document.forms['RQSDefaultForm'].submit();
 }

function button3_onclick() {
 window.open("http://mr-chan:8080/AVXSIntegrationAppWeb/web/UserHelp/rqpuserhelp.html",
"","status=no,toolbar=no,menubar=no,location=no,scrollbars=yes, resizable=yes");
}
//-->
</SCRIPT>
</HEAD>

<BODY bgcolor="#DBECDC" background="http://mr-
chan:8080/AVXSIntegrationAppWeb/web/images/wlbk4.jpg">
<TABLE border="1" width="100%" height="100%" cellpadding="4" cellspacing="1">
 <TR>
<!--Blue Panel Begins Here. To remove the blue panel remove this section-->
  <TD valign="top"  width="2%" height="100%" bgcolor="#3399CC">
 <DIV id="div1" style="DISPLAY: block; WIDTH: 20%" align=center>
   <TABLE border="0"  border="0" align="center">
    <TR align=center><TD>
     <A href="http://www.crystalpoint.com" target="_blank">
      <IMG src="http://mr-chan:8080/AVXSIntegrationAppWeb/web/images/Portal_logo_dark.gif"
border="0" Alt="AppViewXS">
     </A>
    </TD></TR>
    <TR><TD>
     &nbsp;
    </TD></TR>
    <TR align="center"><TD>
    &nbsp;
    </TD></TR>
    <TR><TD>
     &nbsp;
    </TD></TR>
    <TR align="center"><TD>
       <FORM id="form5" name="form5" Action=""><INPUT type="hidden" name="rqs_pageID"
id="rqs_pageID" value="1" >
<INPUT type="hidden"  name="rqs_sessionID" id="rqs_sessionID" value="4ADD52C7377D4BD4" >
<INPUT type="hidden"  name="rqs_custom_dir" id="rqs_custom_dir" value="AVXS6530App" >

      <INPUT type="button" value="Enter" id="rqs_actionID" name="rqs_actionID" onclick="return
customSubmit('Enter')" style="WIDTH: 80%">
      <BR>

      <!--  Sample Additional Button
      <INPUT type="button" value="F1" id="rqs_actionID" name="rqs_actionID" onclick="return
customSubmit('F1')" style="WIDTH: 80%">
      -->

      <BR>
      <INPUT type="button" value="Help" id="button3" name="button3" onclick="return
button3_onclick()" style="WIDTH: 80%">
       </FORM>
    </TD></TR>
   </TABLE>
 </DIV>
  </TD>
<!--Blue Panel Ends Here. Stop Here-->

<!--DO NOT ERASE: Host code and Presentation begins here-->
  <TD width="90%">

<FORM name="RQSDefaultForm" id="RQSDefaultForm" method="post" action="http://mr-
chan:8080/AVXSIntegrationAppWeb/appviewxs">

<TABLE style="FONT-FAMILY: 'courier new',monospace;FONT-SIZE: 12px;" cellspacing="0"
cellpadding="0" border="0">
<TR>
```

```
<FONT face="'courier new',monospace"
size="1"><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</T
D><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&
nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</
TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>
&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;<
/TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD
>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;
</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><T
D>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp
;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><
TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbs
p;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD>
<TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD></FONT></TR><TR>
<TD colspan="80"><FONT face="'courier new',monospace" size="1" style="color: #000000;"><LABEL
id="LABEL1">WELCOME&nbsp;TO&nbsp;k1000.cp.crystalpoint.com&nbsp;[PORT&nbsp;$ZTC0&nbsp;#1047&nbsp;
WINDOW&nbsp;$ZTNT.#PTZLAJ6]</LABEL></FONT></TD>
<TD>&nbsp;</TD>
</TR>
<TR>
<TD colspan="80"><FONT face="'courier new',monospace" size="1" style="color: #000000;"><LABEL
id="LABEL2">TELSERV&nbsp;-&nbsp;T9553D40&nbsp;-&nbsp;(10SEP99)&nbsp;-
&nbsp;(IPMADD)</LABEL></FONT></TD>
<TD>&nbsp;</TD>
</TR>
<TR>
<TD colspan="80"><FONT face="'courier new',monospace" size="1" style="color: #000000;"><LABEL
id="LABEL5">Available&nbsp;Services:</LABEL></FONT></TD>
<TD>&nbsp;</TD>
</TR>
<TR>
<TD colspan="80"><FONT face="'courier new',monospace" size="1" style="color: #000000;"><LABEL
id="LABEL7">TACL&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;EXIT</LABEL></FONT></TD>
<TD>&nbsp;</TD>
</TR>
<TR>
<TD colspan="80"><FONT face="'courier new',monospace" size="1" style="color: #000000;"><LABEL
id="LABEL8">Enter&nbsp;Choice&gt;&nbsp;tacl</LABEL></FONT></TD>
<TD>&nbsp;</TD>
</TR>
<TR>
<TD colspan="7"><FONT face="'courier new',monospace" size="1" style="color: #000000;"><LABEL
id="LABEL9">TACL&nbsp;1&gt;</LABEL></FONT></TD>
<TD>&nbsp;</TD>
<TD colspan="73"><INPUT type="text"    name="TEXT10" id="TEXT10" maxlength="72" value=""
size="72" style="FONT-FAMILY: 'courier new',monospace;FONT-SIZE: 12px;color: #0000ff;background-
color: #ffffff;WIDTH: 511px;" tabindex="1"></TD>
</TR>
<TR>
<TD colspan="81"><INPUT type="text"    name="TEXT11" id="TEXT11" maxlength="80" value=""
size="80" style="FONT-FAMILY: 'courier new',monospace;FONT-SIZE: 12px;color: #0000ff;background-
color: #ffffff;WIDTH: 567px;" tabindex="2"></TD>
</TR>
<TR>
<TD colspan="81"><INPUT type="text"    name="TEXT12" id="TEXT12" maxlength="80" value=""
size="80" style="FONT-FAMILY: 'courier new',monospace;FONT-SIZE: 12px;color: #0000ff;background-
color: #ffffff;WIDTH: 567px;" tabindex="3"></TD>
</TR>
<TR>
<TD colspan="8"><INPUT type="text"    name="TEXT13" id="TEXT13" maxlength="7" value="" size="7"
style="FONT-FAMILY: 'courier new',monospace;FONT-SIZE: 12px;color: #0000ff;background-color:
#ffffff;WIDTH: 56px;" tabindex="4"></TD>
<TD colspan="73">&nbsp;</TD>
</TR>
<TR>

<TD colspan="11">&nbsp;</TD>
<TD colspan="21"><INPUT type="submit" name="LOGON_BUTTON" id="LOGON_BUTTON" value="Logon"
tabindex="27" style="FONT-FAMILY: 'courier new',monospace;FONT-SIZE: 12px;background-color:
#c0c0c0;WIDTH: 146px;"></TD>
<TD colspan="49">&nbsp;</TD>
```

```
</TR>
</TABLE>
<INPUT type="hidden"  name="rqs_pageID" id="rqs_pageID" value="1" >
<INPUT type="hidden"  name="rqs_sessionID" id="rqs_sessionID" value="4ADD52C7377D4BD4" >
<INPUT type="hidden"  name="rqs_custom_dir" id="rqs_custom_dir" value="AVXS6530App" >
<INPUT type="hidden"  name="rqs_functionID" id="rqs_functionID" value="">
<INPUT type="hidden"  name="rqs_activefield" id="rqs_activefield" value="TEXT10" >
<INPUT type="hidden" name="rqs_virtual_signature" id="rqs_virtual_signature"
value="2731276029703109874O" >
<HR><FONT face="'courier new',monospace" size="1" style="color: #000000;"><LABEL
id="RQS_HOST_MESSAGE_STATUS">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;CONV</LABEL></FONT><HR>
<TABLE width="100%"><TR valign="top"><TD width="15%">
<TABLE width="100%" cellpadding="0" cellspacing="0"><TR><TD><SELECT name="rqs_keypadID"
id="rqs_keypadID"  style="FONT-FAMILY: 'courier new',monospace;FONT-SIZE: 12px;">
        <OPTION value="Enter" SELECTED>Enter</OPTION>
        <OPTION value="S-Enter">S-Enter</OPTION>
        <OPTION value="Refresh">Refresh</OPTION>
        <OPTION value="Reset">Reset</OPTION>
        <OPTION value="H-Reset">H-Reset</OPTION>
        <OPTION value="S-Reset">S-Reset</OPTION>
        <OPTION value="Break">Break</OPTION>
        <OPTION value="Exit">Exit</OPTION>
        <OPTION value="F1">F1</OPTION>
        <OPTION value="F2">F2</OPTION>
        <OPTION value="F3">F3</OPTION>
        <OPTION value="F4">F4</OPTION>
        <OPTION value="F5">F5</OPTION>
        <OPTION value="F6">F6</OPTION>
        <OPTION value="F7">F7</OPTION>
        <OPTION value="F8">F8</OPTION>
        <OPTION value="F9">F9</OPTION>
        <OPTION value="F10">F10</OPTION>
        <OPTION value="F11">F11</OPTION>
        <OPTION value="F12">F12</OPTION>
        <OPTION value="F13">F13</OPTION>
        <OPTION value="F14">F14</OPTION>
        <OPTION value="F15">F15</OPTION>
        <OPTION value="F16">F16</OPTION>
        <OPTION value="SF1">SF1</OPTION>
        <OPTION value="SF2">SF2</OPTION>
        <OPTION value="SF3">SF3</OPTION>
        <OPTION value="SF4">SF4</OPTION>
        <OPTION value="SF5">SF5</OPTION>
        <OPTION value="SF6">SF6</OPTION>
        <OPTION value="SF7">SF7</OPTION>
        <OPTION value="SF8">SF8</OPTION>
        <OPTION value="SF9">SF9</OPTION>
        <OPTION value="SF10">SF10</OPTION>
        <OPTION value="SF11">SF11</OPTION>
        <OPTION value="SF12">SF12</OPTION>
        <OPTION value="SF13">SF13</OPTION>
        <OPTION value="SF14">SF14</OPTION>
        <OPTION value="SF15">SF15</OPTION>
        <OPTION value="SF16">SF16</OPTION>
        <OPTION value="RollUp">RollUp</OPTION>
        <OPTION value="RollDn">RollDn</OPTION>
        <OPTION value="S-RollUp">S-RollUp</OPTION>
        <OPTION value="S-RollDn">S-RollDn</OPTION>
        <OPTION value="PageUp">PageUp</OPTION>
        <OPTION value="A-PgUp">A-PgUp</OPTION>
        <OPTION value="PageDn">PageDn</OPTION>
        <OPTION value="A-PgDn">A-PgDn</OPTION>
        <OPTION value="DelLine">DelLine</OPTION>
        <OPTION value="InsLine">InsLine</OPTION>
        <OPTION value="EndFld">EndFld</OPTION>
        <OPTION value="EndPg">EndPg</OPTION>
        <OPTION value="EolErs">EolErs</OPTION>
```

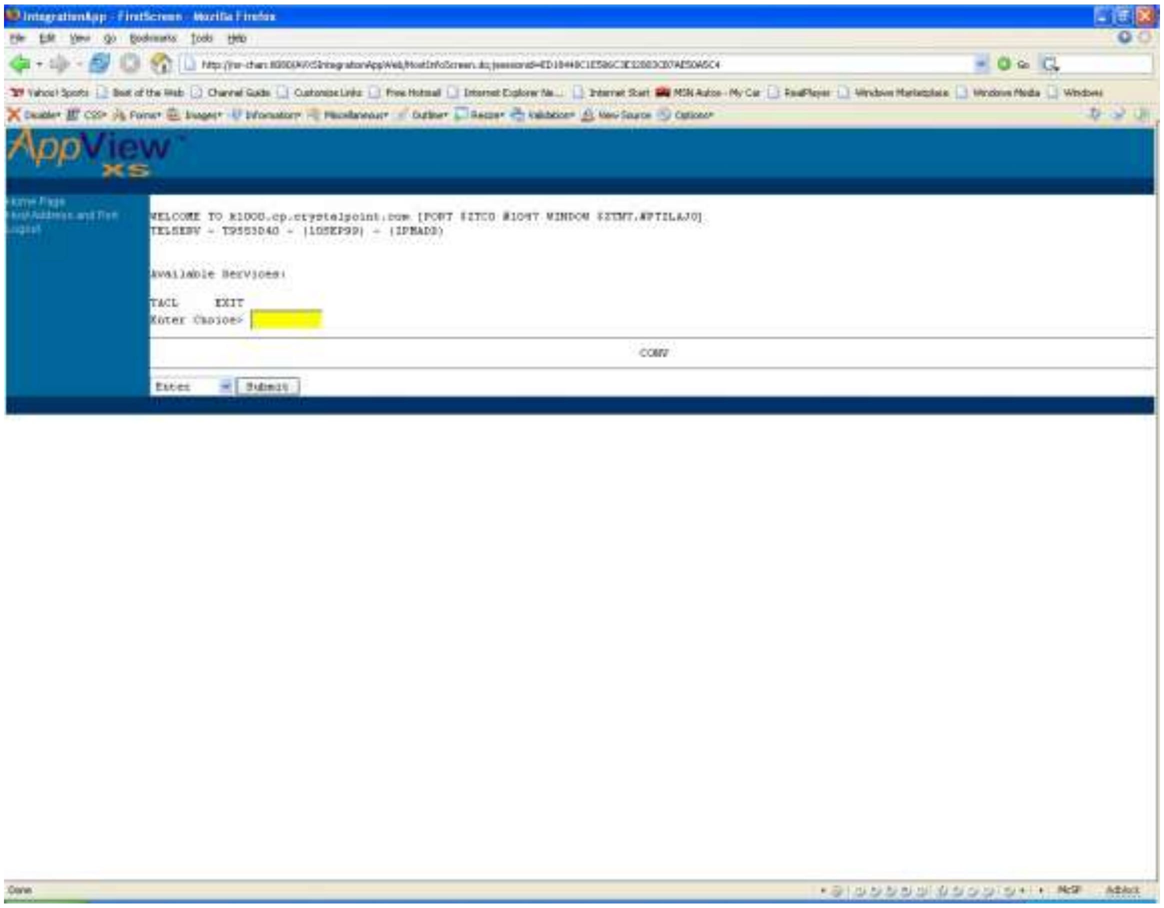
```
       <OPTION value="EopErs">EopErs</OPTION>
</SELECT></TD>
<TD><INPUT type="submit" name="RQSSubmit" id="RQSSubmit" value="Submit" style="FONT-FAMILY:
'courier new',monospace;FONT-SIZE: 12px;WIDTH: 63px;"></FONT></TD>
<TD>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</TD>
</TR>
</TABLE></TD>
<TD>
<TABLE cellpadding="0" cellspacing="0"><TR>
<TD><INPUT type="submit" name="rqs_actionID" id="rqs_actionID" value="REFRESH" style="FONT-
FAMILY: 'courier new',monospace;FONT-SIZE: 12px;WIDTH: 63px;"></FONT></TD><TD><INPUT
type="submit" name="rqs_actionID" id="rqs_actionID" value="ENTER" style="FONT-FAMILY: 'courier
new',monospace;FONT-SIZE: 12px;WIDTH: 63px;"></FONT></TD><TD><INPUT type="submit"
name="rqs_actionID" id="rqs_actionID" value="EXIT" style="FONT-FAMILY: 'courier
new',monospace;FONT-SIZE: 12px;WIDTH: 63px;"></FONT></TD><TD
colspan="5">&nbsp;</TD></TR></TABLE></TD>
</TR>
</TABLE>
<INPUT type="hidden"  name="RQS_BGACTIVE" id="RQS_BGACTIVE" value="#ffff00" >
<INPUT type="hidden"  name="RQS_TXTACTIVE" id="RQS_TXTACTIVE" value="#000000" >

<TABLE width="100%"><TR><TD align="left"><TR>
<TD align="left"><FONT face="'courier new',monospace"  size="1" style="color:
#000000;">Screen&nbsp;ID:&nbsp;</FONT><FONT face="'courier new',monospace" size="1"
style="color: #000000;">logon_tacl</FONT></TD>
<TD align="right"><FONT face="'courier new',monospace"  size="1" style="color:
#000000;">Thank&nbsp;you&nbsp;for&nbsp;using&nbsp;AppViewXS</FONT></TD>
</TR>
</TABLE>
</FORM>
<INPUT type="hidden" id="crystalpoint">
  </TD>
<!--DO NOT ERASE: Host code and Presentation ends here-->
 </TR>
</TABLE>

<SCRIPT LANGUAGE=javascript src="http://mr-
chan:8080/AVXSIntegrationAppWeb/web/js/ncbaseN66530.js">
</SCRIPT>
</BODY>
</HTML>
```

The highlighted red content represents the parsed out data that is used to display to the users via a JSP page. The JSP page will already have the matching HTML tags to accommodate the contents that have been parsed out. When parsing the HTML content, a good starting point is to capture the data inside the <FORM> tags for ‘RQSDefaultForm’. Then you will also need to include the:

`<INPUT type="hidden" id="crystalpoint">` and

```
<SCRIPT LANGUAGE=javascript src="http://mr-
chan:8080/AVXSIntegrationAppWeb/web/js/ncbaseN66530.js">
</SCRIPT>
```

Here is an example of the same screen above, but displayed using JSP with the parsed out date and source:



```
WELCOME TO k1000.cp.crystalpoint.com [PORT $ZTC0 #1047 WINDOW $ZTMT.#PTILAJ0]
TELSERV - T9553D40 - (10SEP99) - (IPMADS)


Available Services:

TACL     EXIT
Enter Choice>
```

The HTML source code is generated by the JSP (appviewxs.jsp) file. The appviewxs.jsp file contains the following content:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<LINK rel="stylesheet" href="theme/document.css" type="text/css">
<META http-equiv="Content-Style-Type" content="text/css">
<META name="GENERATOR" content="IBM WebSphere Studio">
<%@ include file="/include/title.inc" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<SCRIPT type="text/javascript">
// Used to set Action value on Form for AppView screens
var AVXSScreenURL = 'xs<%=titleScreenID%>.do?'+queryString;

</SCRIPT>
</HEAD>
<BODY onload="AVXSIntegrationOnload('RQSDefaultForm');" onunload="AVXSIntegrationOnunload();">
<%-- **** JSP Page include tags go here **** --%>
<%@ page import="avxsintegration.forms.AppViewXSActionForm"%>

<TABLE width="100%" cellpadding="0" cellspacing="0">
```

```
        <TR>
            <TD colspan="2"><%@include file="/include/header.inc" %></TD>
        </TR>
          <TBODY>
          <TR>
               <TD valign="top" width="160" bgcolor="#006699"><%@include
file="/include/nav.inc" %></TD>
                <TD valign="top" width="100%">
                    <html:errors property="hostaccessproblem"/>
                    <html:form name="RQSDefaultForm" type="AppViewXSActionForm" scope="request"
action="xs">
                         <% AppViewXSActionForm appViewForm = (AppViewXSActionForm)
request.getAttribute("RQSDefaultForm");%>
                         <%=appViewForm.getHtmlBody()%>
                    </html:form>
                 </TD>
               </TR>
        </TBODY>
</TABLE>
<TABLE width="100%" cellpadding="0" cellspacing="0">
          <TR>
          <TD height="20" colspan="2" valign="top" class="footer"></TD>
          </TR>
</TABLE>
</BODY>
</HTML>
```

Now the HTML source generated by the appviewxs.jsp after parsing the pertinent data:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<LINK rel="stylesheet" href="theme/document.css" type="text/css">
<META http-equiv="Content-Style-Type" content="text/css">
<META name="GENERATOR" content="IBM WebSphere Studio">

<TITLE>IntegrationApp - FirstScreen</TITLE>
<SCRIPT type="text/javascript" src="script/common.js"></SCRIPT>
<SCRIPT type="text/javascript">
var queryString = 'window='+window.name;
// Used to set Action value on Form for migrated screens
function setFormAction() {
   var x = 'FirstScreenScreen.do?'+queryString;
   document.forms["FirstScreen"].action=x;
}
var global_session_timeout = 30*60*1000; // default 30 minutes
var global_heartbeat = 300*1000; // send heartbeat every 5 minutes
</SCRIPT>


<SCRIPT type="text/javascript">
// Used to set Action value on Form for AppView screens
var AVXSScreenURL = 'xsFirstScreen.do?'+queryString;

</SCRIPT>
</HEAD>
<BODY onload="AVXSIntegrationOnload('RQSDefaultForm');" onunload="AVXSIntegrationOnunload();">



<TABLE width="100%" cellpadding="0" cellspacing="0">
        <TR>
            <TD colspan="2"><div id="heartbeatWindow"
style="position:absolute;width:155;height:65;cursor:hand;left:500;top:0;visibility:hidden;z-
index:9;background-color:#FFFFEE">
       <iframe id="heartbeatFrame" width="155" height="65" frameborder="1" marginwidth="0"
marginheight="0" scrolling="NO">
```

```
        </iframe>
</div>
<TABLE BORDER="0" class="header" CELLSPACING="0" CELLPADDING="0" WIDTH="100%">
    <TR>
        <td width="150">
               <img border="0" width="200" height="55" alt="Company's LOGO"
src="/AVXSIntegrationAppWeb/theme/AppView%20XS%20logo.gif">
        </td>
        <td>
        </td>
    </TR>
    <TR>
        <TD valign="middle" align="center" bgcolor="#003366">
               &nbsp;
        </TD>
        <td>
        </td>
    </TR>
</TABLE>
</TD>
        </TR>
               <TBODY>
               <TR>
               <TD valign="top" width="160" bgcolor="#006699"><TABLE class="nav_top4" border="0"
CELLPADDING="0" CELLSPACING="0" WIDTH="160">
        <TR valign="TOP">
               <TD>
                       <A href="EntryScreen.do" >Home Page</A><BR>
                       <!--
                               <A href="xsFirstScreen.do" >AppView Screen 1</A><BR>
                       -->
                       <A href="HostInfoScreen.do" >Host Address and Port</A><BR>
                       <A href="xsLogOffScreen.do" >Logout</A><BR>
               </TD>
        </TR>
        <TR valign="top">
               <TD valign="top" width="160">&nbsp;</TD>
        </TR>
</TABLE>
</TD>
                <TD valign="top" width="100%">

                       <form name="RQSDefaultForm" method="post"
action="/AVXSIntegrationAppWeb/xs.do">

                                       <TABLE style="FONT-FAMILY: 'courier new',monospace;FONT-
SIZE: 13px;" cellspacing="0" cellpadding="0" border="0">
<TR>
<FONT face="'courier new',monospace"
size="2"><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</T
D><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&
nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</
TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>
&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;<
/TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD
>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;
</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><T
D>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp
;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><
TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbs
p;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD>
<TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD></FONT></TR><TR>
<TD colspan="80"><FONT face="'courier new',monospace" size="2" style="color: #000000;"><LABEL
id="LABEL1">WELCOME&nbsp;TO&nbsp;k1000.cp.crystalpoint.com&nbsp;[PORT&nbsp;$ZTC0&nbsp;#1047&nbsp;
WINDOW&nbsp;$ZTNT.#PTZLAJH]</LABEL></FONT></TD>
</TR>
<TR>
<TD colspan="80"><FONT face="'courier new',monospace" size="2" style="color: #000000;"><LABEL
id="LABEL2">TELSERV&nbsp;-&nbsp;T9553D40&nbsp;-&nbsp;(10SEP99)&nbsp;-
&nbsp;(IPMADD)</LABEL></FONT></TD>
</TR>
```

```
<TR>
<TD colspan="80"><FONT face="'courier new',monospace"  size="2">&nbsp;</FONT></TD>
</TR>
<TR>
<TD colspan="80"><FONT face="'courier new',monospace"  size="2">&nbsp;</FONT></TD>
</TR>
<TR>
<TD colspan="80"><FONT face="'courier new',monospace" size="2" style="color: #000000;"><LABEL
id="LABEL5">Available&nbsp;Services:</LABEL></FONT></TD>
</TR>
<TR>
<TD colspan="80"><FONT face="'courier new',monospace"  size="2">&nbsp;</FONT></TD>
</TR>
<TR>
<TD colspan="80"><FONT face="'courier new',monospace" size="2" style="color: #000000;"><LABEL
id="LABEL7">TACL&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;EXIT</LABEL></FONT></TD>
</TR>
<TR>
<TD colspan="13"><FONT face="'courier new',monospace" size="2" style="color: #000000;"><LABEL
id="LABEL8">Enter&nbsp;Choice&gt;</LABEL></FONT></TD>
<TD>&nbsp;</TD>
<TD colspan="10"><INPUT type="text"    name="TEXT9" id="TEXT9" maxlength="9" value="" size="9"
style="FONT-FAMILY: 'courier new',monospace;FONT-SIZE: 13px;color: #0000ff;background-color:
#ffffff;WIDTH: 80px;" tabindex="1"></TD>
<TD colspan="56">&nbsp;</TD>
</TR>
</TABLE><INPUT type="hidden"  name="rqs_pageID" id="rqs_pageID" value="1" >
<INPUT type="hidden"  name="rqs_sessionID" id="rqs_sessionID" value="D993806A36C84066" >
<INPUT type="hidden"  name="rqs_custom_dir" id="rqs_custom_dir" value="AVXS6530App" >
<INPUT type="hidden"  name="rqs_functionID" id="rqs_functionID" value="">
<INPUT type="hidden"  name="rqs_activefield" id="rqs_activefield" value="TEXT9" >
<INPUT type="hidden" name="rqs_virtual_signature" id="rqs_virtual_signature"
value="26282865117332342985" >
<HR><FONT face="'courier new',monospace" size="2" style="color: #000000;"><LABEL
id="RQS_HOST_MESSAGE_STATUS">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;CONV</LABEL></FONT><HR><SELECT
name="rqs_keypadID" id="rqs_keypadID" style="FONT-FAMILY: 'courier new',monospace;FONT-SIZE:
13px;">
       <OPTION value="Enter" SELECTED>Enter</OPTION>
       <OPTION value="S-Enter">S-Enter</OPTION>
       <OPTION value="Refresh">Refresh</OPTION>
       <OPTION value="Reset">Reset</OPTION>
       <OPTION value="H-Reset">H-Reset</OPTION>
       <OPTION value="S-Reset">S-Reset</OPTION>
       <OPTION value="Break">Break</OPTION>
       <OPTION value="Exit">Exit</OPTION>
       <OPTION value="F1">F1</OPTION>
       <OPTION value="F2">F2</OPTION>
       <OPTION value="F3">F3</OPTION>
       <OPTION value="F4">F4</OPTION>
       <OPTION value="F5">F5</OPTION>
       <OPTION value="F6">F6</OPTION>
       <OPTION value="F7">F7</OPTION>
       <OPTION value="F8">F8</OPTION>
       <OPTION value="F9">F9</OPTION>
       <OPTION value="F10">F10</OPTION>
       <OPTION value="F11">F11</OPTION>
       <OPTION value="F12">F12</OPTION>
       <OPTION value="F13">F13</OPTION>
       <OPTION value="F14">F14</OPTION>
       <OPTION value="F15">F15</OPTION>
       <OPTION value="F16">F16</OPTION>
       <OPTION value="SF1">SF1</OPTION>
       <OPTION value="SF2">SF2</OPTION>
       <OPTION value="SF3">SF3</OPTION>
       <OPTION value="SF4">SF4</OPTION>
       <OPTION value="SF5">SF5</OPTION>
       <OPTION value="SF6">SF6</OPTION>
```

```
        <OPTION value="SF7">SF7</OPTION>
        <OPTION value="SF8">SF8</OPTION>
        <OPTION value="SF9">SF9</OPTION>
        <OPTION value="SF10">SF10</OPTION>
        <OPTION value="SF11">SF11</OPTION>
        <OPTION value="SF12">SF12</OPTION>
        <OPTION value="SF13">SF13</OPTION>
        <OPTION value="SF14">SF14</OPTION>
        <OPTION value="SF15">SF15</OPTION>
        <OPTION value="SF16">SF16</OPTION>
        <OPTION value="RollUp">RollUp</OPTION>
        <OPTION value="RollDn">RollDn</OPTION>
        <OPTION value="S-RollUp">S-RollUp</OPTION>
        <OPTION value="S-RollDn">S-RollDn</OPTION>
        <OPTION value="PageUp">PageUp</OPTION>
        <OPTION value="A-PgUp">A-PgUp</OPTION>
        <OPTION value="PageDn">PageDn</OPTION>
        <OPTION value="A-PgDn">A-PgDn</OPTION>
        <OPTION value="DelLine">DelLine</OPTION>
        <OPTION value="InsLine">InsLine</OPTION>
        <OPTION value="EndFld">EndFld</OPTION>
        <OPTION value="EndPg">EndPg</OPTION>
        <OPTION value="EolErs">EolErs</OPTION>
        <OPTION value="EopErs">EopErs</OPTION>
</SELECT><INPUT type="submit" name="RQSSubmit" id="RQSSubmit" value="Submit" style="FONT-
FAMILY: 'courier new',monospace;FONT-SIZE: 13px;WIDTH: 72px;">
<INPUT type="hidden" id="crystalpoint">
 <SCRIPT LANGUAGE=javascript src="http://mr-
chan:8080/AVXSIntegrationAppWeb/web/js/iebase6530.js">
</SCRIPT>
                    </form>
                        </TD>
                      </TR>
        </TBODY>
</TABLE>
<TABLE width="100%" cellpadding="0" cellspacing="0">
               <TR>
               <TD height="20" colspan="2" valign="top" class="footer"></TD>
        </TR>
</TABLE>
</BODY>
</HTML>
```

Depending on what you want displayed to your end users, you can parse out the contents that are pertinent to your users. The specific contents that are still needed would be the contents that include the hidden input fields. These hidden variables are used to store specific screen constants that are used by the AppViewXS Struts logic and the AppViewXS emulation sessions. There are many hidden input field variables, but the ones that are most important are:

- rqs_pageID
- rqs_sessionID
- rqs_virtual_signature

When replying to the host, these data are needed to keep the session in sync with the AppViewXS emulation. There are other hidden input variables such as:

- rqs_activefield
- rqs_actionID
- rqs_functionID

Even though these hidden fields are not critical for the emulation to function, they do provide useful information to the AppViewXS emulation session. For example, rqs_activefield tells us which field on the browser is the active field when the emulation response is displayed in the browser, rqs_actionID tells the emulator wich function key the user pressed on the keyboard, and the rqs_functionID is use to depict which drop down list function key the user selected. These hidden input variables are all accessible by javascript.

Using Struts to Provide the Plugin Point to AppViewXS 2.01 Application

Struts is an open source framework for building Java web applications. It is used to help extend and create a plugin point to control access to AppViewXS 2.01 application. The development of the plugin point is quite straightforward if you are knowledgeable in using the Struts framework. Those familiar with Struts will readily understand the benefits in using this framework to provide a convenient plugin point to access AppViewXS 2.01.

With Struts, a single Servlet is usually all that is required to support all of the URLs for a single web application. The struts-config.xml file maps the URL to a specific Struts Action class to handle the request. Typically the response is then forwarded on to a JSP for the final composition of the user interface.

A simple way of looking at using the Struts framework in AppViewXS 2.01 is to use a single Business logic class, a single helper class to isolate the communication with AppViewXS and a single JSP to render the output for all AppViewXS screens.

Business Logic (Struts Action) Class

The business logic class, also known as the Struts Action class, represents the plugin point in which the class (avxsintegration.actions.AppViewXSDispatcherAction) is responsible for communicating via a helper class (avxsintegration.helper.AppViewXS Proxy) with the AppViewXS application and then forwarding the HTML returned from AppViewXS to the corresponding JSP file. You can use this Action class to leverage your existing Security and infrastructure modules to control access into AppViewXS 2.01.

You can also control the HTML data returned from the AppViewXS 2.01 application and parse it to achieve a standard User Interface. For instance, if the COBOL screens all displayed a Date/Time stamp on every screen, but the Web Application already supports a JavaScript clock on the header of every page, the Action class may parse the HTML and strip out the fields corresponding to the clock returned by AppViewXS 2.01. This is one of the most powerful reasons to ‘get between’ the user and AppViewXS. It allows for a standardization of the final HTML rendering across all screens without creating screen-specific maps in AppViewXS Studio.

The Action class can also be used to facilitate the synergy between an AppViewXS application and an external web or java application. After processing the data, the modified data can either be passed onto AppViewXS to transmit to the NonStop Host or posted back to the external application.

The Action class provides a convenient point for users to ‘hook up’ AppViewXS 2.01 to become an integral part of their overall product solution.

Helper Class

The helper class (AppViewXS Proxy) is used to pass the request to AppViewXS 2.01 application via the J2EE RequestDispatcher class. The ‘include’ method passes the request to the AppViewXS Servlet and then returns the generated response. All communication between the original web application and the AppViewXS is accomplished via a single stateful Java class. This class will parse the returned HTML response and store some of the values of the hidden HTML fields returned by AppViewXS. The minimum sets of value to store in this helper class are:

- rqs_pageID
- rqs_sessionID
- rqs_virtual_signature

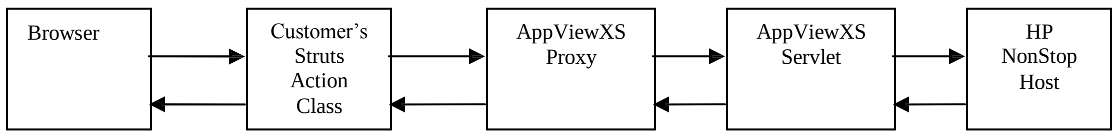
These values must be provided in all future communication with AppViewXS. The helper class then returns the HTML response from the NonStop host back to the Business class which is mapped to a JSP file for rendering.

Single JSP to Render all AppViewXS 2.01 Screens

A single JSP that contains the layout which is consistent with the look and feel of all the other screens is created. For many Web Applications this means that a common header will be displayed on every screen along with a navigation menu. The JSP will retrieve the parsed HTML response from the Action class via a Struts ActionForm and display it in the same location as all other screens use. The AppViewXS HTML response has already been parsed to remove the <HEAD> portion so that the HTML that is present in the ActionForm may be directly embedded into the JSP.

Simple Flow Chart of the Whole Process

To clarify the picture, it often helps to have a diagram to depict the flow of information in AppViewXS 2.01



The Action class provides the access point to external applications and other re-usable infrastructure related modules. The AppViewXS Proxy is responsible for maintaining the connection specific information as well as specific error handling to the AppViewXS servlet. The AppViewXS servlet provides the connection access to the HP NonStop Host and returns the HTML response back to the AppViewXS Proxy. AppViewXS Proxy parses the HTML response looking for specific connection information to update its state. The Action class then formats the response to parse the HTML data that should be displayed to the end user.

The avxsintegration Package Source Code

The avxsintegration package contains code that you can use to integrate AppViewXS 2.01 application into your existing web application. The package is subdivided into 5 logical subfolders. These subfolders are:

- actions
- forms
- helper
- resources
- utility

Actions Folder

The actions folder contains the classes that are used to interact with the AppViewXS application and other external applications. This folder is mainly where you would want to place your business logic classes and other functionality that you would like to extend or enhance your existing legacy application. The actions folder currently contains the following source code files:

- AppViewXSDispatcherAction – this class controls the request flow between the browser and AppViewXS application with the help of AppViewXSProxy.
- AppViewXSHearbeatAction – this class is intended to provide a mechanism to send a heart beat to your legacy application so that a time out does not occur.
- AVXSConfigScreenAction – this class contains the logic to perform the AppViewXS application installation.
- EntryScreenAction – this class is a simple STRUTS demo screen that takes two input fields and then combines them into a single name.

- HostInfoAction – this class is a screen that takes the input for host address and port number which is used to by AppViewXS to connect to your host with the specified address and port.
- ZipValidationAction – this class is a simple demo that takes the address information provided by the ZipValidation.jsp and posts the information to a USPS service which provides the zip code plus 4 information..

Forms Folder

The forms folder contains classes that are used to store the input field data from the JSP files. The data stored in these forms are then used by the action classes to perform their functions. The forms folder currently contains the following source code files:

- AppViewXSActionForm – this class contains the HTML data retrieved from an AppViewXS emulation session.
- AVXSConfigScreenActionForm – this class contains the setting information used for the AppViewXS application installation.
- EntryScreenActionForm – this class contains the first and last name information data used by the Struts demo.
- HostInfoActionForm – this class contains the data for the host address and port number.
- ZipValidationForm – this class contains the data used to store the address information.

Helper Folder

The helper folder contains classes that provide utility functions to aid Action classes in their functionality. The helper folder currently contains the following source code files:

- ActionHelper – this class provides a simple utility to retrieve the names of the screen URI and strip off the .do extensions. It is just a simple helper class to identify the screen you are currently on.
- AppViewXSProxy – this class provides the functionality to communicate with the AppViewXS application. It also contains all the specific AppViewXS session information.
- AppViewXSResponseStream – this class is used to cache the output from AppViewXS application for post-processing of the HTML response.
- AppViewXSResponseWrapper – this class is used to buffer the response for post-processing of the HTML emitted by AppViewXS.
- ConfigResult – this class is a simple helper form that is used to store AppViewXS installation specific messages.
- ExternalApplicationProxy – this class provides a simple proxy to interact with external web application or site to post information to and then retrieve the result back.
- IConstants – this class is an interface class used to contain constants that is used by the avxsintegration package.

Resources Folder

The resources folder contains a single file named “ApplicationResources.properties”. This properties file is part of the Struts framework which is used for custom error message handling. This file provides a convenient centralized location to keep all the error messages that gets generated.

Utilities Folder

The utilities folder contains source files that is used to aid in the installation process for the AppViewXS application. All the files in this folder form the global search and replace tool used for the AppViewXS installation. The utilities folder currently contains the following source code files:

- BoyerMoore – this class file contains the algorithm used to perform the search. Note that this source code is adapted from Michael Lecuy.
- CPSearchAndReplace – this class file is the main file that is used to configure and perform the search and replace.
- extensions.txt – this is a sample file containing extensions that can be used to configure the extensions types for CPSearchAndReplace to filter through.
- ParseFile – this class file contains the logic to open up a file and read through the file using the BoyerMoore class to search for the target text and then replace it with the new value.

Incorporating New Functionality

Adding new feature and functionality to AVXSIntegrationAppWeb is relatively straightforward. Working within the Struts framework, you will need to do the following:

1) Create the Action class to perform your specific logic/function.
2) Create the ActionForm bean to store and retrieve data used for the Action class.
3) Add the Action class and ActionForm information into the struts-config.xml file.
4) Create the corresponding JSP file to display UI and handling user interaction.

The example “Zip code look up” link in the AVXSIntegrationAppWeb demo is a simple JSP file that users fill out their address information in order to get their zip code plus 4 information.

Zip Code Validation Demo

Note: This demo assumes that you already know how to customize your screens using the AppViewXS Customization Studio Tool.

This demo works by using the ZipValidation.jsp form to fill in customer address information. The demo requires that you create a custom screen using the AppViewXS Customization Studio tool. Before you begin the customization process for this demo, you will need to create a global variable named "zip_plus4_result". Make sure that you create this variable before you customize your screen as this variable needs to be pre-defined first using the Session Configuration tool of the Customization Studio. After you have pre-defined the "zip_plus4_result" global variable, then you can begin and create a custom screen for the zip validation demo.

The text fields you will need to create will correspond to the following labels:

- Name
- Address
- City
- State

You will also need to create a label to store the zipcode value retrieved from the demo. For the text fields that you created next to the corresponding labels, you will need to configure the global variables to store/write the data that are entered into the text fields. The global variables that are used for this demo corresponding to their text fields are the following:

- Name Text Field – firmGV
- Address Text Field – address1GV
- City Text Field – cityGV
- State Text Field – stateGV

Then next to the label you have created for the zipcode, you will need to create another label which will read from the global variable "zip_plus4_result".

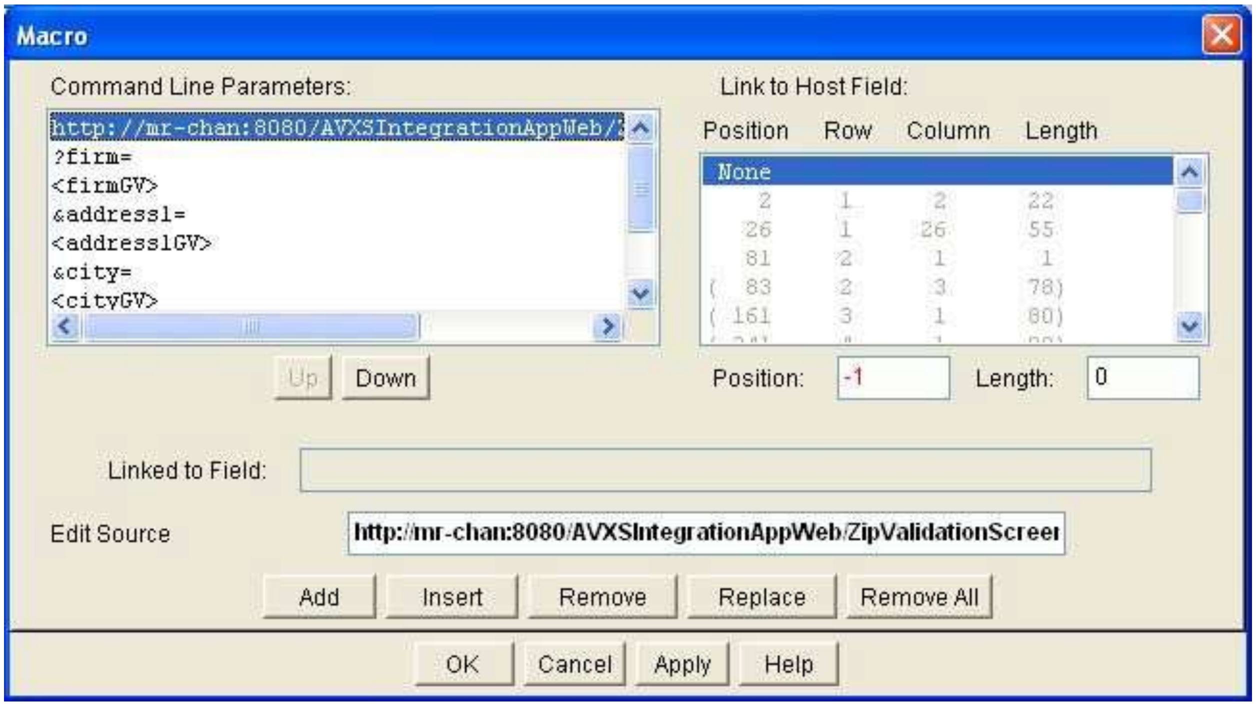
Next, you will need to create a button (label it Store GV) so that the data entered into the text field gets stored into each global variable. This button will need to be assigned the <P_C> behavior.

Finally, a web link is created in which it is customized via the studio tool to post the data stored into the text fields as parameters to the ZipValidation.jsp file. For the created link, using the Studio tool, select the Macro option and enter the following information into the "Command Line Parameters" window:

- http://your web application url/AVXSIntegrationAppWeb/ZipValidationScreen.do?firm=
- <firmGV>
- &address1=

- <address1GV>
- &city=
- <cityGV>
- &state=
- <stateGV>

Here is an example image of the Macro dialogue box for the web link:



Command Line Parameters:

```
http://mr-chan:8080/AVXSIntegrationAppWeb/
?firm=
<firmGV>
&address1=
<address1GV>
&city=
<cityGV>
```

Link to Host Field:

Position	Row	Column	Length
None			
2	1	2	22
26	1	26	55
81	2	1	1
(83	2	3	78)
(161	3	1	80)

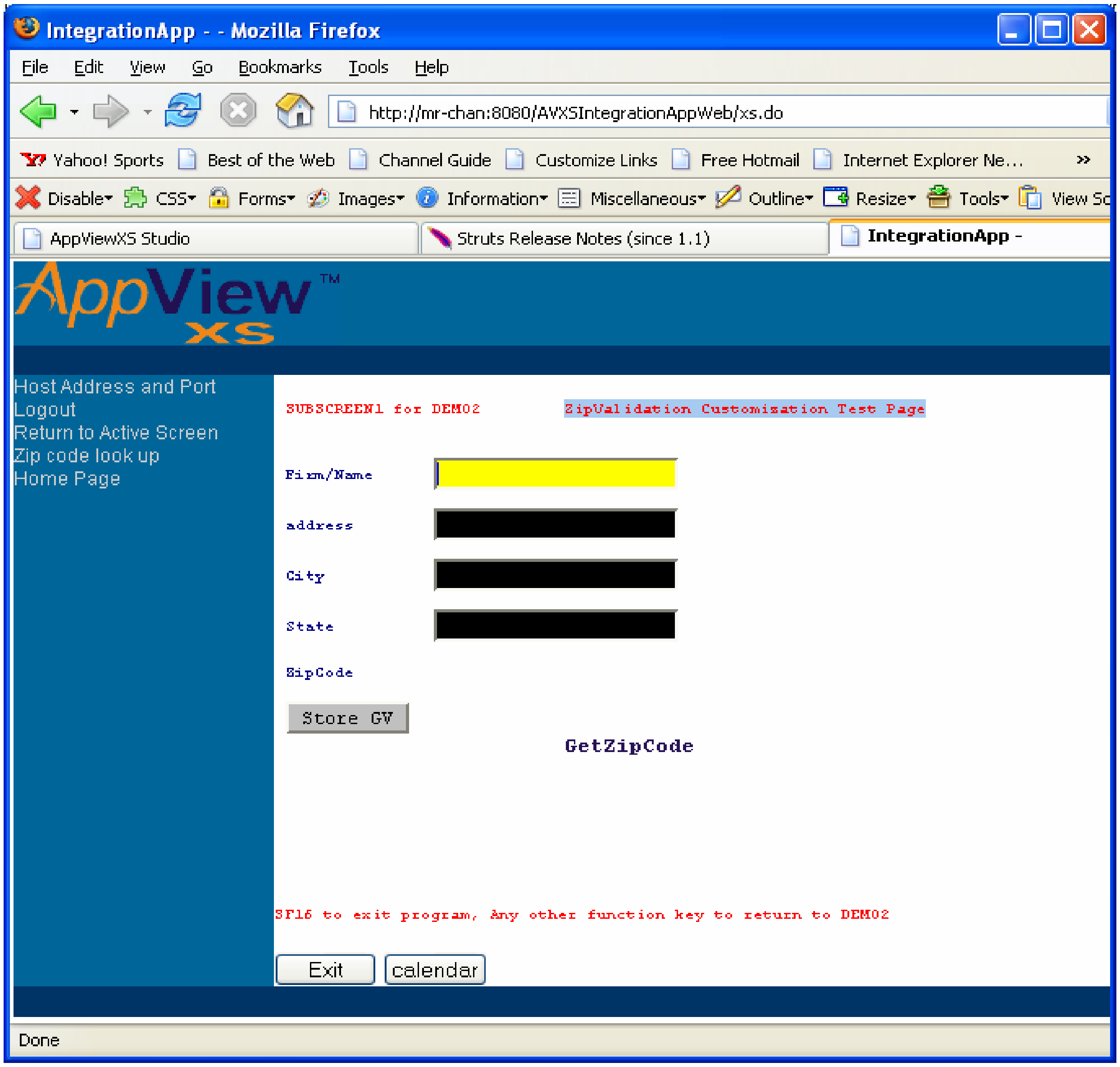
Position: -1 Length: 0

Linked to Field:

Edit Source http://mr-chan:8080/AVXSIntegrationAppWeb/ZipValidationScreen

When the user has entered all the pertinent information into the customized screen, presses the customized button to write the data into the different global variables and then clicks on the link "GetZipCode", the data is passed as parameters to the ZipValidationScreen URL to be processed by the ZipValidation.jsp class.

Here is an example screen customized for the zipcode demo:


IntegrationApp - - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://mr-chan:8080/AVXSIntegrationAppWeb/xs.do
Yahoo! Sports Best of the Web Channel Guide Customize Links Free Hotmail Internet Explorer Ne...
Disable CSS Forms Images Information Miscellaneous Outline Resize Tools View So
AppViewXS Studio
Struts Release Notes (since 1.1)
IntegrationApp -
AppView XS
Host Address and Port
Logout
Return to Active Screen
Zip code look up
Home Page
Store GV
Exit
calendar
Done

SUBSCREEN1 for DEMO2 ZipValidation Customization Test Page

Firm/Name

address

City

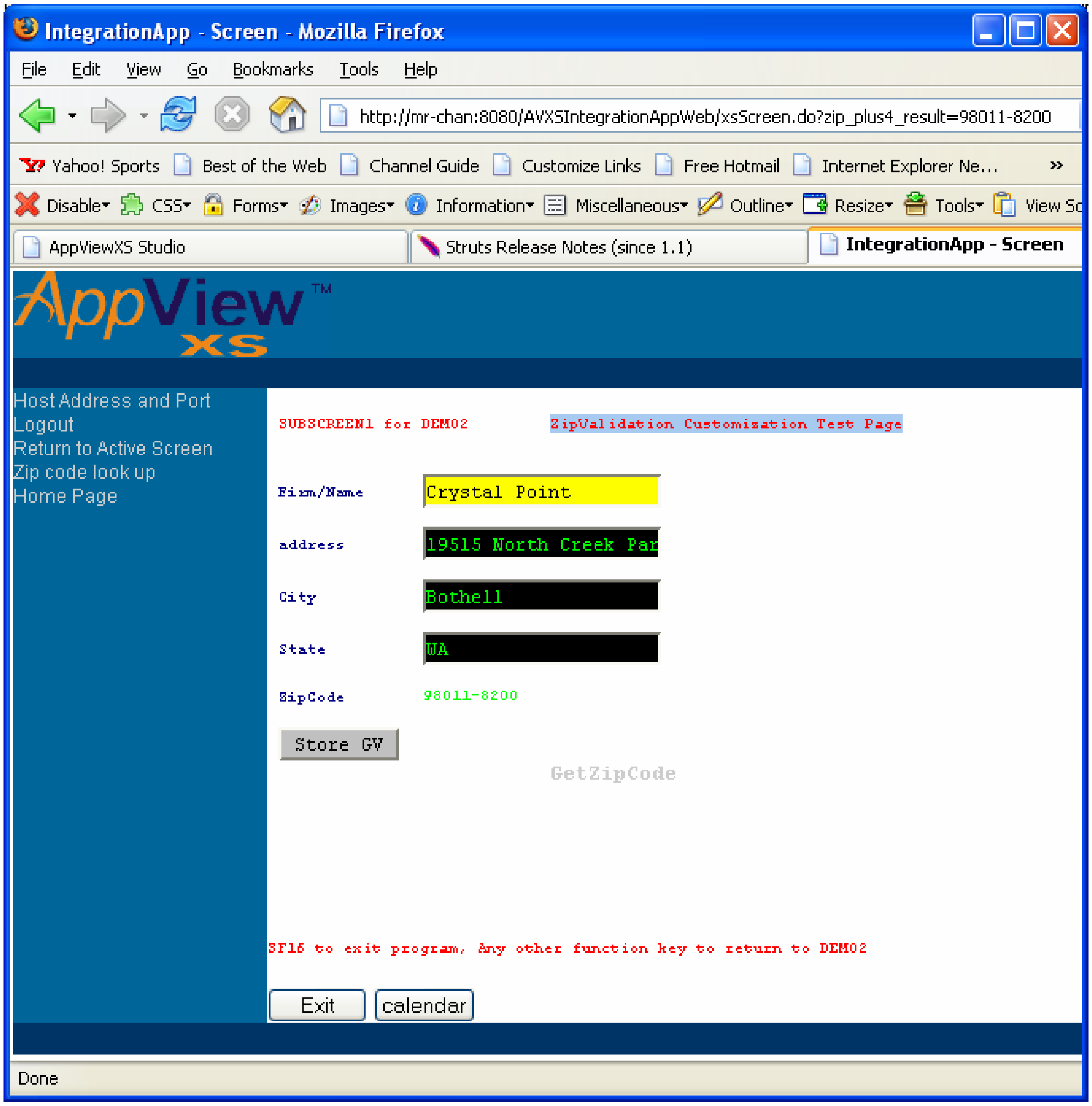
State

ZipCode

GetZipCode

SF16 to exit program, Any other function key to return to DEMO2

When this demo is hooked up from a customized screen such as a web link that reads in the parameters the user had entered via the global variables, the user then clicks on the web link which in turn passes the parameters to the ZipValidation.jsp page. This page having its parameters in the form field filled with valid content will then perform a submit (Post) to the external USPS web application. The result is then stored into the zip_plus4_result field in the Form AddressForm. Then a javascript function is called when the page is reloaded. This javascript code automatically redirects the browser back to the active AppViewXS session with the zip_plus4_result appended to the URL. Here is an example screen with zip code encoded into the URL:


IntegrationApp - Screen - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://mr-chan:8080/AVXSIntegrationAppWeb/xsScreen.do?zip_plus4_result=98011-8200
Yahoo! Sports Best of the Web Channel Guide Customize Links Free Hotmail Internet Explorer Ne...
Disable CSS Forms Images Information Miscellaneous Outline Resize Tools View So
AppViewXS Studio
Struts Release Notes (since 1.1)
IntegrationApp - Screen
AppView XS
Host Address and Port
Logout
Return to Active Screen
Zip code look up
Home Page
Store GV
GetZipCode
Exit
calendar
Done

SUBSCREEN1 for DEMO2 ZipValidation Customization Test Page

Firm/Name	Crystal Point
address	19515 North Creek Par
City	Bothell
State	WA
ZipCode	98011-8200

SF16 to exit program, Any other function key to return to DEMO2

Note that the zip_plus4_result and its value is encoded into the URL string. If you do not want to see the parameters encoded into the URL, you can probably use an invisible form to submit the zip_plus4_result back to the AppViewXS servlet.

Depending on your needs, it is helpful sometimes to create a link (Return to Active Screen) on the navigation panel that brings the emulation session back into focus. Majority of times when you click on a link, you end up losing the emulation session because your browser is now pointed at a different page. If the history back button is disabled, then you would have no way of returning back to your active emulation session.

Creating new external application and also integrating your existing application with your legacy application is not complex, but will require coding on your part to customize the specific details using the existing AppViewXSIntegrationAppWeb frame work as a guideline to enhance and extend your existing legacy application.

Additional AppViewXS Feature

LoadBalance Feature

The load balanced feature of AppViewXS provides a simple Round Robin loadbalancer logic to spread the connections out evenly to different hosts. The host address values are configured in the "loadbalanced_url.cpi" file which is located in the folder:

at2custom/at2wsp folder.

In the "loadbalanced_url.cpi" file, the first entry must list out the number of distinct host addresses that will be in this file. For example, if you have 5 different host URL, then this first entry value will be 5. The variable name used in this file to depict the number of different host address will be "distinctURL". Then the next following entries into the file will contain the string "URL" and a number representing the different host address and port. The variable naming convention will be in a sequential incremental fashion of the numerical digit at the end of the string "URL". Note that the host port should also be included into the host address definition. Be sure to use a colon to seperate the host address and port value.

An example file would look something like this:

```
distinctURL=5

URL1=tn6530://tandem.crystalpoint.com:23
URL2=tn6530://tandem2.crystalpoint.com:23
URL3=tn6530://tandem3.crystalpoint.com:24
URL4=tn6530://tandem4.crystalpoint.com:23
URL5=tn6530://tandem5.crystalpoint.com:22
```

The string "tn6530://" tells the AppViewXS servlet which transport protocol we are using for the emulation session.

If you do not want to use the load balanced feature of this product, you can simply delete the loadbalanced_url.cpi file from the at2wsp folder or comment out every line in the file. To comment out each line, all you need to do is prepend the pound sign ("#") on each line. This will revert the AppViewXS servlet to use address and port setting from the Portal.html or whatever launcher file you use to start an AppViewXS session.

Tunneling Servlet Feature

The tunneling servlet feature of AppViewXS provides a way to have AppViewXS sessions communicate through the Tunneling Servlet to the host. The tunneling servlet is included with the AppViewXS installation CD and is stored in the following CD directory:

zip_files\tunnelservlet.zip

The documentation for the tunneling servlet is also on the CD and is located in:

documentation\AppViewXS Tunneling Servlet Readme1.doc

A new feature only within AppViewXS
is that you can supply two different Tunneling Servlet URL address using AppViewXS's Tunneling servlet config file "tunneling_config.cpi". This new feature provides a back up Tunneling servlet that will be automatically re-directed to the secondary Tunneling servlet address if the primary Tunneling servlet address is not responding. The tunneling servlet address values are configured in the "tunneling_config.cpi" file which is located in the folder:

at2custom/at2wsp folder.

In the "tunneling_config.cpi" file, there are 3 simple entries for configuring the AppViewXS session to communicate with the Tunneling Servlet. These three entries are:

portalTunnelURL1 --	this variable represents the primary URL address to find the tunneling servlet
portalTunnelURL2 --	this variable represents the secondary backup URL address for a backup tunneling servlet
portalTunnelDebug --	this variable turns on or off the trace utitilty found in the AppViewXS session when communicating with the Tunneling Servlet. Note that the files created by the AppViewXS session is located on the machine running the AppViewXS application.

An example file would look something like this:

```
portalTunnelURL1=http://mr-chan:8080/resqtunnelservlet/AppViewTunnel
portalTunnelURL2=http://mr-chan:8080/tunnelservlet/AppViewTunnel
portalTunnelDebug=false
```

If you do not want to use the tunneling servlet feature of this product, you can simply delete the tunneling_config.cpi file from the at2wsp folder or comment out every line in the file. To comment out each line, all you need to do is prepend the pound sign ("#") on each line. This will revert the AppViewXS servlet to not use the Tunneling Servlet.

Jakarta Commons Logging Support

AVXSIntegrationAppWeb uses the Jakarta Commons Logging. Located inside AVXSIntegrationAppWeb’s WEB-INF/classes folder, you should see the log4j.properties file. This file contains the parameters used to configure the logging properties. All instances of the logger are created using the Jakarta Commons LogFactory. Because we also package the log4j-1.2.9 jar file, the default logging created by the LogFactory will be log4j. The log4j.properties file is configured to display messages on the java console. The error message levels are set to the ‘INFO’ level. To reduce the display of messages in the java console, you can change the error message level from INFO to ERROR level. Currently the only locations in the code that has the logging capability are located in the avxsintegration.utility and avxsintegration.helper packages. The other locations did not need to have the logging capability. To learn more about using the Jakarta Commons Logging, you can checkout:

http://jakarta.apache.org/commons/

Adding logging capability into your custom code is easy. Since the library jar files for the Jakarta Commons Logging and the Log4j already exists in the WEB-INF/library folder, you can just call these modules in your code. To start logging you will need to create an instance of a logger. To accomplish this you would add the following line to your code:

```
public class CLASS
{
    private static Log log = LogFactory.getLog(CLASS.class);
    ...
    ;
```

The Class.class represents the name of the class file that you are currently working on and would like to log the output from. The Log class contains 6 various logging levels ranging from trace, debug, info, warn, error, and fatal. Depending on what you want filtered out and displayed on the console, you would use log.info(String message), log.trace(String message), etc… Then using the log4j.properties file, you can enter the level of detail that you would like to see by specifying the error level.