IBM Integrated Web Application Server for i

IBM Systems & Technology Group
April 2009

This document is a ‘work in progress’ and will continue to be updated as time permits.
Preface

This IBM documentation provides information on the IBM Integrated Web Application Server for i (IAS) from the view of an Integrated Solution Vendor (ISV). This paper was written to provide the application developers the tools they need to deploy applications using IAS and to successfully ship and deploy their applications to their customers.

This document is a ‘work in progress’. It will continue to be updated as time permits. If you have comments, suggestions or updates, please feel free to email them to timmr@us.ibm.com. Thanks!

Team that wrote this documentation

**Rich Macrafic** is a Senior Managing Consultant at IBM. He accepted a position at IBM 32 years ago, directly out of college at George Washington University in Washington, D.C. Over his career, he has held various development, management, staff and consulting positions. The last 15 years have been spent in the areas of Web and Object Oriented technologies, in roles ranging from software engineer to architect to consultant.

**Andy Henderson** is an Advisory Software Engineer working in the IBM Rochester Support Center for 12 years. Currently his expertise is in Java and operating system performance.

**Jon Rush** is a technical consultant in the IBM STG ISV Enablement organization. He is a senior software engineer who specializes in WebSphere, IBM Hypertext Preprocessor (PHP) and IP telephony on the System i platform. Jon has helped hundreds of System i solution providers enhance their applications to use IBM e-business technologies such as IBM Net.Data®, WebSphere PHP and Integrated Web Application Server.

**Tim Rowe** is a Senior Software Engineer with responsibilities in Web based middleware in IBM i. He is currently the team lead for the Web Integration team that support the IBM HTTP Server (powered by Apache) for i, Integrated Web Services Server, and IBM Integrated Web Application Server. He help various positions across the IBM i operating system and has spent the past 9 years focused on the Web based technologies.
1. Part I – Integrated Web Application Server

1.1 Overview

1.1.1 Abstract

This document provides an introduction to the IBM Integrated Web Application Server for i (IAS) that ships with version 5 release 4 and version 6 release 1 of the IBM i (formerly i5/OS) operating system.

The audience targeted is the broad base of users needing to host Web-based applications. Included in that target group are the ISVs who want to build and deploy IAS applications to their own customers. As such, this document addresses both the IBM Web Administration for i interface (GUI) and the programmatic approach to working with the IAS product.

For the ISV, much of the benefit of this document is derived from the sample code that is included. That sample code is written with best practices in mind and is well-documented via Javadoc. The rest of this document augments the details and supplements your understanding of the programmatic concepts.

For the audience which is not part of the ISV community, it is possible that much of what you need to know and accomplish can be handled by the GUI. You can, therefore, focus on the GUI portions of this document and let your requirements dictate how much deeper into this resource, if at all, you need to go.

1.1.2 What is IAS?

The Integrated Web Application Server (IAS), included in the IBM i 4.5 and IBM i 6.1 operating system, addresses the need for a minimal footprint, easy to configure, secure infrastructure for hosting Web applications. IAS enables a subset of the full Java™ 2, Enterprise Edition (J2EE) specification; if your application uses from among static web components (HTML, images, docs, etc.), servlets, JavaServer Pages (JSPs) and/or JavaServer Faces (JSF) components, this alternative to the IBM WebSphere® Application Server environment requires less memory resources to deploy and run your application. The added benefit of requiring less memory to run this server makes it attractive on smaller-footprint IBM i hardware.

IAS is built on common programming models and open standards, thus facilitating integration into a broader and varied architecture. Specifically, IAS is based on Open Services Gateway initiative (OSGi) technology. Refer to the white pa-
IBM Integrated Web Application Server for i

per, About the OSGi Service Platform, for more details about OSGi (see Section Appendix A Additional References/Resources).

IAS has been implemented by IBM as the foundation application server for both IBM Systems Director Navigator for IBM i (formerly i5/OS) and IBM DB2® Web Query. When you use those products, you are using IAS!

1.1.3 Philosophy

The primary interface for creating and managing IAS servers, is through the IBM Web Administration for i GUI interface. This is the 'port 2001' interface. Details on how to access are below. The GUI interface is not intended to expose all aspects of the IAS server. The GUI attempts to cover all the basic functions and the majority of the every day items. The more advanced items can be modified via manual manipulation of the environment (property files, xml files, etc) or through the callable SPI interfaces.

1.1.4 Target Application Environment

Here are some of the reasons you may want to use Integrated Web Application Server for i:

- You imbed ASF Tomcat in your application today. With the integrated Web application server you would not have to imbed ASF Tomcat with your application.
- You ran your application on the version of ASF Tomcat shipped with 5722DG1 which in IBM i 6.1 has been removed from service.
- You have limited system resources which prevent you from optimally running WebSphere Application Server
  - The Integrated Web Application Server focused features and functions typically take 40% less memory then WebSphere Application Server. The default for the WebSphere Application Server running on IBM System i 515 Express is 1 GB of memory.
  - There is no additional memory needed for startup for the integrated Web application server. It is recommended to have 2 GB of memory to for startup for WebSphere Application Server.
  - You want to deploy simple Web applications on your server and do not need the cross platform capabilities, additional security, n-tier deployment environment, JMS & EJB programming model support, SOA features in WebSphere Application Server, WebSphere Application Server ND clustering, or multiple JNDI connections.
- You want to deploy simplified database applications with one data base connection.
- You want to deploy traditional servlet and JSP applications.
- Don’t want to have to bother with installing another product (IAS is included in the OS)
- Want to stay with the IBM i fix delivery for obtaining service. IAS is maintained via IBM i PTFs.
1.1.5 Comparisons to other Application Servers

The Integrated Web Application Server compares very favorable to other well know Web based application servers. Due to the power of the OSGI framework the IAS server can take advantage of some of the same components that other servers make use of. For example the ‘Web container’ used by IAS is the exact same Web container that is used by WebSphere. The database pooling code is the same database container that is used by the ASF Tomcat server. The IAS server has been able to leverage pieces from several technologies to produce this versatile and light weight server.

WebSphere Application Server

Use WebSphere Application server if you want the following:

- You need higher availability or more scalability
- You are supporting cross platform deployment and only want to test with one application server
- You use more then one JNDI (Java Naming and Directory Interface) connection
- You want to use the built-in messaging engine based on WebSphere MQ, or the Java Based Messaging Engine
- You need more of the robust features in WebSphere Application Server such as:
  - Support for full J2EE programming model
    - EJB’s, JMS
    - JDBC (IAS has limited namespace, only allows one JNDI connection)
  - Support for full & extended J2EE packaging model
    - WAR, EAR, RAR files
    - IBM extensions (extended deployment descriptor, annotations, etc.)
  - Support for n-tier deployment model
    - HTTP server, application server, DB on same or different machines
    - cross platform, heterogeneous capabilities
      - Run tiers on different servers, operating systems, etc.
      - Support for DB2, Oracle, MySQL, etc. database access
  - Support for full range of security options (HTTP server, Web application, Java)

Tomcat

Since 2000, the IBM i operating system has been shipping and making use of ASF Tomcat version 3.2. This application server has been shipped as part of 5722DG1 product. This version of ASF Tomcat has basically remained unchanged since we shipped it. This application server has a very small memory footprint and works excellent for very simple applications. Support for this application server has been removed in IBM i 6.1. The last release that applications running on this server will work is IBM i 4.5. Where this version of ASF Tomcat
IBM Integrated Web Application Server for i

was never updated on i, there have been many new enhancements and updated
specifications that are now supported by newer versions of ASF Tomcat.

The latest versions of ASF Tomcat have many of the same features and func-
tions that are in more advanced application servers today. Any of the newer ver-
sions of ASF Tomcat can be run successfully on IBM i. You need to go to the
Apache.org Web site and download the version of ASF Tomcat you need. The
process of obtaining the server runtime and maintaining the server are com-
pletely your responsibility to maintain.

To see the different specifications of ASF Tomcat, please review these at the
Apache.org website. For the purpose of this document, we are going focus on
IBM related servers.

With the general view of ASF Tomcat being a small footprint, low memory, simple
application server, the IAS server is a logical server to move towards. (This is
certainly true for any user running their applications on the version of ASF Tom-
cat shipped in the 5722DG1 product.)

1.2 Installation

The Integrated Web Application Server runtime is a part of the IBM i operating
system. It is shipped as part of 5722SS1 option 3 (IBM i 4.5) and 5761SS1 option
3 (IBM i 6.1). Support for this server is delivered via PTFs. Since this application
server is a part of the operating system, there are no additional products that
need to be downloaded or installed. There are no unfamiliar fix methods to be fol-
lowed, all fixes are delivered via normal IBM i PTFs.

1.2.1 Dependent Products

The Integrated Web Application Server runtime is a part of the IBM i operating
system. It does have some dependencies on other products and options of the
operating system in order for the complete solution to work correctly. Below is the
list of IBM i products that are required for the IAS server to work correctly:

**IBM i 6.1**

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Base</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5761SS1</td>
<td>*BASE i5/OS</td>
<td>Extended Base Support</td>
</tr>
<tr>
<td>5761SS1</td>
<td>1</td>
<td>Extended Base Directory Support</td>
</tr>
<tr>
<td>5761SS1</td>
<td>3</td>
<td>Host Servers</td>
</tr>
<tr>
<td>5761SS1</td>
<td>12</td>
<td>Qshell</td>
</tr>
<tr>
<td>5761SS1</td>
<td>30</td>
<td>Portable App Solutions Environment</td>
</tr>
<tr>
<td>5761DG1</td>
<td>*BASE IBM HTTP Server for i5/OS</td>
<td></td>
</tr>
<tr>
<td>5761JV1</td>
<td>*BASE IBM Developer Kit for Java</td>
<td></td>
</tr>
<tr>
<td>5761JV1</td>
<td>8</td>
<td>J2SE 5.0 32 bit</td>
</tr>
</tbody>
</table>
IBM Integrated Web Application Server for i

**IBM i 5.4**

<table>
<thead>
<tr>
<th>Product Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5722SS1</td>
<td>*BASE      i5/OS</td>
</tr>
<tr>
<td>5722SS1</td>
<td>1          Extended Base Support</td>
</tr>
<tr>
<td>5722SS1</td>
<td>3          Extended Base Directory Support</td>
</tr>
<tr>
<td>5722SS1</td>
<td>12         Host Servers</td>
</tr>
<tr>
<td>5722SS1</td>
<td>30         Qshell</td>
</tr>
<tr>
<td>5722SS1</td>
<td>33         Portable App Solutions Environment</td>
</tr>
<tr>
<td>5722DG1</td>
<td>*BASE      IBM HTTP Server for i5/OS</td>
</tr>
<tr>
<td>5722JV1</td>
<td>*BASE      IBM Developer Kit for Java</td>
</tr>
<tr>
<td>5722JV1</td>
<td>8          J2SE 5.0 32 bit</td>
</tr>
</tbody>
</table>

**1.2.2 PTFs**

Due to the product dependencies and to make the delivery of fixes easier to get and maintain, the Group PTF process is heavily used. If you keep 2 group PTFs up to the most current levels you can always to be sure to be at the latest maintenance levels. For both IBM i 6.1 & 5.4 the IAS server was shipped as part of a group PTF. Below are the Group PTF numbers for each release along with the recommended minimal levels.

**IBM i 6.1**

<table>
<thead>
<tr>
<th>PTF Number</th>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF99115</td>
<td>5</td>
<td>IBM HTTP SERVER FOR I</td>
</tr>
<tr>
<td>SF99562</td>
<td>5</td>
<td>JAVA</td>
</tr>
</tbody>
</table>

**IBM i 5.4**

<table>
<thead>
<tr>
<th>PTF Number</th>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF99114</td>
<td>15</td>
<td>IBM HTTP SERVER FOR I</td>
</tr>
<tr>
<td>SF99291</td>
<td>17</td>
<td>JAVA</td>
</tr>
</tbody>
</table>

**1.2.3 Install Directory Structure**

The IAS server is installed to a ‘proddata’ location, like many IBM i products. The install location is ‘/QIBM/Proddata/OS/OSGI/LWI71’. It is important that files under this path are not modified in anyway. If files under this path are changed, then all IAS servers on the system will be changed. This is likely not the desired behavior.
The default location for created instances is the ‘www’ directory. When an IAS server is created, select key files and directories are created under the server name directory, and then all the actual runtime files are symbolically linked back to the proddata location. Be sure that only files under the server creation directory are modified.

1.3 GUI Interface

1.3.1 Accessing the GUI

The IBM Web Administration for i (Web Admin) interface provides an easy to use and access interface for creating and managing all of your Web related servers on the i. Web Admin allows you to see all your Web related servers. You can see and manage, IBM HTTP Servers powered by Apache, IBM WebSphere Application Server, IBM WebSphere Portal, and the new IBM Integrated Web Application Server for i.
Figure 2 - IBM Web Administration for i Setup page

The Web Admin GUI is shipped with the operating system, part of the 5761DG1 & 5722DG1 products. The GUI interface runs within the ADMIN servers and once these are running is completely accessible via your favorite browser.

**Note:** in order to use the Web Admin GUI interface, your user profile will require the following special authorities:

*IOSYSCFG and *ALLOBJ

Yes, we are aware this authority level is not possible in all circumstances. In the future this restriction will be removed.

**Starting the ADMIN servers**

Before you can access the Web Admin GUI you need to start the ADMIN servers. To verify the current status of the ADMIN servers, sign on to a green screen window and enter the following:

```
WRKSBJSJOB QHTTPSVR
```

Verify that the ADMIN jobs are active.
If the ADMIN server jobs are not active, start the admin jobs using the following command:

\texttt{STRTCPSVR SERVER (*HTTP) HTTPSVR(*ADMIN)}

**Accessing the Web Admin GUI**

The Web Admin GUI is accessible using your favorite Web browser.

**Note:** The main path when accessing the GUI has changed between IBM i 5.4 and IBM i 6.1. The differences are outlined below.

**IBM i 4.5**

Enter the following URL to get to the i5/OS Task Page.

\texttt{http://<hostname>:2001}
IBM Integrated Web Application Server for i

Figure 4 - I5/OS Tasks page

From the Task page click on the IBM Web Administration for i5/OS link to launch into the Web Admin GUI interface.

IBM i 6.1

Enter the following URL:

http://<hostname>:2001

You will be redirected to the IBM Systems Director Navigator for i5/OS interface. Systems Director Navigator is the new system interface for the IBM i. This interface allows you to perform many of the functions you have done in the ‘green screen’ or the IBM Operations Navigator PC client interface in the past. This new interface is shipped as part of the operating system and requires no additional software to be installed. There is now over 300 IBM i tasks that are available for use within this new Web based interface. This new interface also runs as part of the ADMIN servers.

Click ‘OK’ on the Security Domain warning.

Figure 5 - Security warning Firefox
IBM Integrated Web Application Server for i

Figure 6 - Security warning Internet Explorer

Signon to the Director Navigator interface using your IBM i user profile.

Figure 7 - IBM Systems Director Navigator interface

This will now launch you into the Welcome page of the new IBM Systems Director Navigator interface. There are a couple of different options you can take to get to the Web Admin GUI interface.

Option 1 - Click on the i5/OS Tasks link in the middle of the right work pane.
IBM Integrated Web Application Server for i

Figure 8 - IBM Systems Director Navigator Welcome

This will launch you to the i5/OS Task page where you can click on the IBM Web Administration GUI link.

Figure 9 - i5/OS Task page
IBM Integrated Web Application Server for i

Option 2 – In the left navigation area click on Internet Configuration. In the Internet Configuration right pane, click on the IBM Web Administration link.

Figure 10 - Internet Configuration page

One you have click on the IBM Web Administration GUI link, you will again be prompted for a user profile. Signon using your IBM i user profile. This will launch you into the Web Admin GUI interface.

Figure 11 - Web Administration GUI Signon

Access IBM Web Administration GUI directly

The Web Admin GUI interface is also directly URL accessible. To avoid these interfaces mentioned above, enter the following URL directly into you Web browser. The URI is case sensitive, so be sure to enter as specified below.

http://<hostname>:2001/HTTPAdmin
1.4 Instance Creation and Configuration

1.4.1 Creating an IAS instance

Creation Via the GUI

This section is here just for reference. From the ISV perspective, the programmatic solution is the likely end destination, using the GUI for testing and development can be a far more convenient solution.

Access the GUI and sign in using an IBM i user profile that has *IOSYSCFG and *ALLOBJ private authorities. **Note:** we realize this level of elevated authority is not possible in all cases and will be looking to remove this restriction in the future.

When creating an IAS server, you are basically creating an application server. The IAS server is just one of the application server options available on the IBM i. In the left-hand pane of the GUI, under the Common Tasks and Wizards heading, select the Create Application Server link. If you are on the Setup main tab, the Create a New Application Server link is also available in the middle of the main work area.

![Create a New Application Server link](image)

**Figure 12 - Create a New Application Server link**

The first page of the Create Application Server wizard is the Introduction page. This page is here really for the first time user. It gives some detail about what
IBM Integrated Web Application Server for i

makes up an application server and what pieces this wizard will be creating and configuring. Read through the information and click the Next button to continue.

![Create Application Server - Introduction panel](image)

Figure 13 - Create Application Server - Introduction panel

The Create Application Server wizard is for creating all the different types and versions of application servers that are available on IBM i. This includes the WebSphere Application Server and the Integrated Web Application Server. The Version panel displays all the different types and versions of application servers that have been installed on this system. All the different version of WebSphere Application Server will show along with the IAS server version. Select the IAS server radio button, this will be the top most link in the right work area pane.
Once you have selected the type of serve to create, you are prompted for the name and description for this server. The name of the server must be any alphanumeric value that is unique for this server type. A name is defaulted for you, but it is often best to specify a meaningful name to help identify this server later. Specifying a meaningful description can also help identify this server.
IBM Integrated Web Application Server for i

Click the Next button to proceed. The server ports panel is the next piece of information that is required. The ports field is not a single port, but rather a range of ports that the server will use. An application server uses several ports during its processing. The server listens on one port for requests that it is to process, other ports are used for communication within the server. The Create Application Server wizard recommends a range of ports that are free. Ensuring the ports in the range are free is very important. A port can only be used by a single server or service at a time. If the port specified for this server is already in use when this server is attempting to start, the server start-up will fail. The wizard will make every attempt to ensure that the ports specified are free. The wizard checks all the currently active ports on the system, all the HTTP Apache server, WebSphere Application Server, and Integrated Web Application Servers that are configured on this system.

Figure 16 - Create Application Server - Server ports

Specify the ports that you want the server to use, or accept the ports recommended by the wizard and click the Next button to continue. The port specified is the first port in the range. The wizard will make use of the next 10 ports following the specified port.

The ‘front door’ to the application server is the IBM HTTP Server (powered by Apache). The HTTP server is created for several reasons. It provides an additional level of protection between your system and the Web. If you have concerns about the security for your application, you can easily configure the HTTP server to only accept requests on SSL. The HTTP server is also used to easily stop and start the servers. The 2 servers are tied together. This allows you to use the
IBM Integrated Web Application Server for i

‘STRTCPSVR/ENDTCPSVR’ commands to start and stop both of these servers together.

Specify the unique name for this HTTP server, the IP address and port this HTTP server is to use. The IP Address for the HTTP server can listed on a specific IP Address or hostname or can be configured to listed on all IP addresses.

Figure 17 - Create Application Server - HTTP Server

The next panel is where you specify the user profile for this IAS server to run with. When this server is started, this user profile will be used as the runtime user profile. This means that this server will only be able to access items that the runtime user profile has authority to. This is an important security control for the server. The default profile for the IAS server is QLWISVR. This user profile has no special authority. If you wish to specify your own user profile, select the Specify an existing user ID option. If the user ID you are currently signed into the GUI with has *SECADM special authority, the Create new User ID option will be available. This option will create a new user profile with the specified name for use by this IAS server. This new user ID is created with *USER special authority.
Figure 18 - Create Application Server - Specify Runtime User ID

The last panel for the Create Application server wizard is the Sample Application panel. For the IAS server, we don’t have a special sample application that is installed, but rather there is a function that shipped as part of the IAS server that can serve a similar purpose. The Eclipse Help interface, can be used a way to test to see if the IAS server is working correctly. This is a great tool to help determine if there is a problem with one of your deployed applications, or is there some problem where the routing or the server is not working correctly. If you can access the Eclipse Help page, then it is likely a problem with your application as the routing and IAS server would be running. Click Next to continue.
The Summary panel provides a detailed overview of the values that specified in the wizard along with the values defaulted by the wizard. In the bottom right hand corner, there is a Print Summary button that allows you to see all these values in a printer friendly manner.
Click *Finish* to start the process of creating the IAS server. This process will run in the background. Your browser will be directed to the Manage server page. The IAS server will remain in the *Creating* status until the process complete. For an IAS server this is normally a very quick process. Once the server is created, it will automatically start.

![manage_server.png](image)

**Figure 21 - Manage server - Server Creating**

Once the server is *Running* you can proceed with deploying application and running the deployed applications. To verify the server is working correctly, enter the URL for the ‘help’ application as mentioned on the Summary page. Forgot to write that URL down? No problem, Click on the *View Create Summary* link to view the information displayed on the create server summary page.

**Programmatically**

See the javadoc for the ISV Redbook Sample Code for details and code examples on creating a server programmatically.

**1.4.2 Directory Structure**

By default, the directories created with each IAS Instance include (Not complete list)

/www/<IAS_InstanceName>/lwi
IBM Integrated Web Application Server for i

This lwi directory contains all of the IAS related configuration files and logs. Several of the objects in this directory include symbolic links.

```
Figure 22 - Server root path /www/<IAS_InstanceName>/lwi

/www/<IAS_InstanceName>/lwi/logs
```

This logs directory contains the diagnostic log files related to the IAS server.

```
Figure 23 - Server Logs path /www/<IAS_InstanceName>/lwi/logs

/www/<IAS_InstanceName>/lwi/apps/eclipse/plugins
```

The plugins directory contains both symbolic links to bundles in the /QIBM/Proddata/OS/OSGi/shared directory, as well as the deployed application.
IBM Integrated Web Application Server for i

Figure 24 - Server Applications Location path

/www/<IAS_InstanceName>/lwi/conf/overrides/

The overrides contains property files that define the instance. It is important that only the files in overrides are changed, rather than changing files that exist right off of the conf directory, as that can have negative impact on the other IAS instances.

Figure 25 - Server Configuration path
1.4.3 Ports

There are a possible four ports created with each IAS server. These ports are configured in the following IAS server configuration file:

/www/<IAS_InstanceName>/lwi/conf/overrides/config.properties

Three of these ports are directly configured for each IAS server. In the config.properties file you can find the following port parameters:

com.ibm.pvc.webcontainer.port - This is the internal HTTP port used for the server. This port can be used in addition to, or in place of the Apache port.

com.ibm.pvc.webcontainer.port.secure - If SSL is configured, this port will be configured as the port that is used for secure SSL connections to the server.

com.ibm.lwi.admin.http.port - This port is used for administrative purposes and internal use only.

The fourth port that is configured for each IAS is the port used for the front end HTTP Server. This port is configured in the Apache server configuration file. The port is defined by the Listen directive that is located in the /www/<IAS_InstanceName>/conf/httpd.conf file.

When an IAS server is created, there is a create file that is created that defines all the values that were used to create this server. This is valuable for service and can help you verify what was originally specified when the server was created vs any updates and changes that have been done since the servers creation. The create serve properties file is located at: /www/<IAS_InstanceName>/iasCreate<IAS_InstanceName>.properties

Please note this file is only for service purposes. Any changes to values in this file have no impact to the actual runtime for the IAS server instance.

Several important values in this file are:

com.ibm.lwi.instance.apache.port - Same port that is specified in the Apache server configuration file httpd.conf

com.ibm.lwi.instance.web.admin.port - port to the administration tools for the IAS server

com.ibm.lwi.instance.web.port - the main Web port for the IAS server

com.ibm.lwi.instance.web.securePort - the SSL port when configured for the IAS server.
1.4.4 Property/XML Files

Configuration changes are made in property and xml files. Below are a list of important files and their description.

/www/<IAS_InstanceName>/conf/  This directory contains property files that control the IAS server startup process, the configuration file for the front-end HTTP Server, and the plugin routing files.

i5OSStartup.properties  This file determines the startup values for the IAS server. Properties include lwiVersion, javaVersion, serverroot, stdout and more. This file also controls the IBM i job description and subsystem that the IAS server will be started in.

httpd.conf  This file defines the configuration for the front-end Apache server. All valid Apache server directives can be specified for this Apache server. This file contains the Apache server port (defined by the listen directive) and the plugin routing module and plugin file location.

lwi-plugin-cfg.xml  This XML file contains the routing information that allows requests to be safely handled by the Apache server and then directed to the IAS server for processing. This file contains the Apache port and the corresponding IAS server port that is listing for requests. The plugin file also defines the URI values that the IAS server are to process. By default when IAS and Apache server is created, ALL URI requests are sent to the IAS server for processing. If you want to have the Apache server process some static server content, this plugin file will need be updated to specify the specific URI values that the IAS server should be processing. This will then allow the Apache server to process all other requests.

/www/<IAS_InstanceName>/lwi/conf/overrides/  This overrides directory contains properties files that can be changed to affect the IAS_Instance. Below are several of the most important:

i5javaopts.javaopt  This file contains java properties. Initial heap, Max Heap, and JVM arguments can be specified here. If you have some additional important JVM argument required, this is the place to specify that option.

logging.properties  Logging specifications can be set in this file. JSR47, RCP, and RCP tracing is configured here. Section 4.2 goes into logging detail.

sec.properties  Authorization and authentication is configured in this file.

database.properties  Database connection properties are configured here. DB Name, Type and connection information can be stored in this file. Section 3 goes into further detail the database configuration.

config.properties  IAS configuration details are stored in this file. This file determines the location of other .properties files.

More information on properties and XML files can be found in Appendix D.
1.4.5 Notes

If problems encountered during instance create, the server is left in an unknown (probably unusable) state. Recovery is to remove the server and start over.

1.5 Apache Http Server

The Apache server is created as the ‘front end’ to the Web environment. Having the HTTP server created provides several useful options.

- The HTTP server

1.5.1 Directory Structure

The apache configuration by default resides in /www/<IAS_InstanceName>/conf. In this conf directory, is the httpd.conf file, where the main HTTP configuration exists.

The http specific logs are in /www/<IAS_InstanceName>/logs.

1.5.2 Remote Web Server

The IAS server can be front ended with any Apache server. The http server can either be local or can be on a remote system.

1.5.3 Notes (TBD)

GUI and programmatic handling of listener I/P addresses (defaults to ‘*’)

[Placeholder for Rob’s Document]

1.6 Runtime Environment

The IAS server is a highly customizable server environment. There are many aspects of the job that can be modified. Since the server is integrated into the IBM i operating system, there are both operating system and server customizations that can be made. The server runs within a JVM that is started in a IBM i job. This allows that job to be controlled in a similar manner as other IBM i jobs.
1.6.1 Subsystem Jobs

The jobs that make up the IAS server currently run as part of the QHTTPSVR subsystem. This is the default location for many of the Web based servers that run on IBM i. This is something that is changeable, but does require some knowledge of the IBM i operating system.

The file that controls the start properties of the IAS server is

`<Instance Root>/conf/i5OSStartup.properties`

The important values in this file are the properties that begin with 'startup'. The default JobQueue, Class, and Routing Data are all located in the QHTTPSVR library. To change the subsystem, update these job control values to your corresponding values and start the server.

![i5OSStartup.properties file](image)

Figure 26 - i5OSStartup.properties file

1.6.2 Storage Pools

The default subsystem for IAS servers is the QHTTPSVR/QHTTPSVR subsystem. This subsystem run by default in the system *BASE memory pool.
The memory pool for the subsystem is specified by the pool definition values for the subsystem description. To change the memory pool for a subsystem, you need to first decide if you want to run in a Share pool or a private pool.
IBM Integrated Web Application Server for i

Private memory pool
Identified by subsystem name in iSeries Navigator, it is a pool in which a single subsystem can run jobs. Private pools are pools of main storage that cannot be shared by multiple subsystems. A private pool contains a specified amount of storage to be used by only one subsystem. A private pool requires no additional configuration. The private pool is specified on the subsystem description. A private pool size and activity level cannot be changed by the performance adjuster (QPFRADJ).

Shared memory pool
A shared memory is a pool in which multiple subsystems can run jobs. Using shared memory pools allows the system to distribute similar jobs across multiple subsystems, still allowing these jobs to run in the same memory pool. A shared memory pool also allows the System i performance adjuster (QPFRADJ) to automatically change the size and activity level.

To view the pool definitions for your subsystem enter the work with subsystem description command for your subsystem.

WRKSBSD SBSD(lib/sbs_name)

Enter option 5 to Display the subsystem information. From the display subsystem description panel select option 2 Pool definitions.
Display Subsystem Description

Subsystem description: TIMMRSBS   Library: TIMMR
Status: ACTIVE

Select one of the following:

1. Operational attributes
2. Pool definitions
3. Autostart job entries
4. Work station name entries
5. Work station type entries
6. Job queue entries
7. Routing entries
8. Communications entries
9. Remote location name entries
10. Prestart job entries

Selection or command

Figure 29 - Display subsystem description

The Display Pool Definitions screen will display showing the current memory pool definitions for this subsystem. Make note of the pool ID number. This number will be important for the next step.

Display Pool Definitions

Subsystem description: QHTTPSVR   Status: ACTIVE

<table>
<thead>
<tr>
<th>Pool ID</th>
<th>Storage Size (K)</th>
<th>Activity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>#BASE</td>
<td></td>
</tr>
</tbody>
</table>

Figure 30 - Display Pool Definition
IBM Integrated Web Application Server for i

Now that you have the current memory pool definition for the subsystem you can change the value using the CHGSBSD command.

Setting the memory pool to a Private pool

Using the change subsystem description command, setting a private pool is very simple. Specify the pool ID number, the size of the memory pool, and the maximum activity level. Enter CHGSBSD lib/sbs_name and hit the Pf4 key to prompt on the command.

```
Figure 31 - Change Subsystem Description command - Private Pool definition
```

In the example above, we are setting a private pool in pool identification 1 (was the *BASE memory pool) creating a memory pool of 1000 Kilobytes, with a maximum activity level of 400. What does all this mean? This will replace the existing pool ID number 1 values with these newly specified values. When this subsystem is started, 1000 kilobytes will be removed from the *BASE pool and will be placed in this private pool. **Note:** this is an important thing to remember, creating a private pool can potentially improve the individual performance of the Web server running in the private pool, but the memory put in this memory pool will be removed from the *BASE pools allotment of memory. This can potentially have a negative impact on other aspects of system performance.

1.6.3 Profiles

The IAS server is made up of a number of different ‘profiles’ that when combined provide a level of functionality. One of the core design goals for IAS has been to
IBM Integrated Web Application Server for i

enable a “right-sized” and customized runtime environment. To meet the varied needs, functionality in IAS is delivered across a set of profiles. A profile is comprised of a set of IAS features and associated plug-ins, providing it with a defined capability set and resource requirements.

The IBM i team has done its best to hide this level of function from the users, by creating the IAS server with a ‘normal’ set of profiles that provide the function that all external deployers will require to successfully create and run an application. The default profiles are uimin, database, extended.

Following are short descriptions of the profiles available in IAS. NOTE: By default, IAS instances are created with uimin, database and extended profiles.

**core** – The IAS Core Platform profile includes the Eclipse Core, which provides OSGi and Extension Point Framework support. This profile also serves to support security (including authentication) and plug-in lifecycle customization support. This is the minimal profile to create basic services and is a requisite profile to all other profiles.

**uimin** – The UI Min profile provides the necessary enablement to run Servlet/JSF-based applications.

**database** – The Database profile provides services to utilize a default database (Apache Derby) and helper services to communicate to a variety of databases, including DB2, Oracle and Microsoft SQL.

**extended** – This profile provides support for JMS and EJB support. It is the most functionally rich profile and is a composite profile that aggregates the services from a number of the other profiles.

**nonstop** - The NonStop profile provides the functionality to ensure reliability and availability of the IAS platform. The runtime consists of two servers; the defined IAS server and a monitor daemon, that run independently of one another. Each verifies the other's runtime status, restarting the other, if required.

**uimax** – A User Interface profile, the UI Max profile provides additional User-interface toolkits and extensions, such as AUIML.

**uistandard** – Another User Interface profile, the UI Standard profile provides the necessary runtime to host portlet-based applications.

**webservicesmin** – The WebServices Profile contains the necessary runtime to host web services. Support for stateless web services is based on an augmented version of the JSR 172 specification. Support for stateful services is initially compliant with WSRF1.2 specification, but is expected to evolve toward WSDM 1.0/WS-RT over time.

The information on these profiles is a more advanced feature that is likely not going to be needed by most all ISV solutions. For example, the Web Administration GUI creates the IAS servers with a basic set of profiles for all applications. If you do have a need to customize these, you will need to do this through the programmable interfaces.
2. Part II – Applications

2.1 Application Development

2.1.1 Environments

RAD
Rational Application Developer (RAD) is an integrated development environment (IDE) used for visually designing, testing and deploying web services, portlets and Java 2 Enterprise Edition (J2EE) applications.


RDi
IBM Rational Developer for i (RDi) is a new eclipse-based workstation offering for IBM i (i5/OS) application development. It provides an IDE for creating IBM i RPG, COBOL, and CL-based applications and is based on the latest Eclipse 3.4.1 level.


WDSC
Websphere Development Studio Client (WDSC) provides an IDE for developing Java, Web, i5/OS server applications using languages such as RPG and COBOL. WDSC is the predecessor to RDi.

http://www-01.ibm.com/software/awdtools/wdt400/

2.1.2 Dependent Bundles
Something needs to be added here

2.2 Application Packaging

2.2.1 WAB File

Traditional J2EE applications are packaged as WAR or enterprise archive (EAR) files. The IAS instance requires a package that is called a Web application bun-
IBM Integrated Web Application Server for i

dle (WAB). These bundles are based on the OSGi R4 Framework for Web applications. An OSGi bundle comprises Java classes and other resources, which, together, provide classes for the Web application. The System i Web administration function generates a JAR file (referred to as a WAB) from your WAR file. This bundle contains the following resources and information:

- **Resources to implement the Web application**: These resources can be class files for the Java programming language, as well as other data, such as HTML files, JSP files, help files and icons.

- **A manifest file**: This file describes the contents of the JAR file and provides information about the bundle. The manifest file uses headers to specify parameters that are needed to install and activate a bundle.

- **Dependencies on other resources**: These dependencies, such as Java packages, must be available to the bundle before the bundle can run. The dependencies for these packages are resolved prior to starting a bundle.

- **A special class in the bundle**: This class acts as the bundle activator. It is instantiated and invoked to start and stop methods, which, in turn, are used to start or stop the bundle. Cleanup operations can be performed when the bundle is stopped.

At a minimum, you will see the artifacts in the WAB file as portrayed by this screenshot:

![Figure 32 - Bundle JAR (WAB) contents](image)

### 2.2.2 Manifest File

The MANIFEST file exists in the META-INF directory of the WAB (.jar) file. This manifest file is generated as part of the WAB .jar file generation. The MANIFEST.MF file contains data that is needed but the framework to correctly install and activate the application. In general, you do not need to modify this MANIFEST.MF file. Most of the contents of the MANIFEST.MF files is created...
IBM Integrated Web Application Server for i

when the WAR file is converted to the WAB file. There are a few key elements to the MANIFEST.MF that you must pay attention to. The key manifest options include:

- **Bundle-Activator**: This class is used to start and stop the bundle. The default class is the org.osgi.service.webapplication.WebApplicationActivator class, which extends org.eclipse.core.runtime.Plugin which, in turn, implements the BundleActivator interface to start the Web application in the IAS instance.

- **Bundle-ClassPath**: This property specifies the CLASSPATH to use for the bundle. The property can contain references to directories or JAR files that are inside the bundle jar file. You can use the period (.) to indicate the bundle's root. The default is a single directory, WEB-INF/classes.

- **Bundle-Version**: This property specifies the version number of the bundle. Package imports and the required bundle specifications can include a bundle version number.

- **Import-Package**: This property specifies all the packages to explicitly import from the required plug-ins. By default, all packages must be resolved for a bundle to start. You can also specify package imports as being optional — to support cases in which a package might not exist. Explicitly imported classes are resolved before packages from Require-Bundle plug-ins are resolved. **Note**: this is a very important element, this is the one element that is often not able to be completely generated by the WAR to WAB conversion. If your application has dependencies on external jars, these Jar must be defined in the import list. Most all NoClassDefFound exceptions that occur are due to a missing jar in the import-package section.

- **Require-Bundle**: This property specifies which bundles (as well as their exported packages) to import for use in the given bundle. The specified bundles are evaluated after performing the explicit package imports.
2.2.3 Generating the WAB

Since the SPI bundle (com.ibm.i5OS.lwiadmin) is not normally a part of a user’s IAS instance, to be able to access the SPI classes above, we must create a symbolic link (symlink) the SPI bundle in the applications plugins directory, after the application is installed.

```bash
strqsh
cd /www/<server>/lwi/apps/eclipse/plugins
ln -s /QIBM/ProdData/OS/OSGi/shared/com.ibm.i5OS.lwiadmin com.ibm.i5OS.lwiadmin
```

The above ln –s is not necessary if PTF SI34549 is applied. This ptf will create this symbolic link automatically.

If you were so inclined, you could study the OSGi specifications in depth and manually construct your own WAB files. Fortunately, we have tools available to us that allow us to avoid such an inconvenience.

Both Rational Application Developer (RAD) and WebSphere Development Studio Client for System I (WDSCI) allow us to generate a war file that is usable for our purpose. Of note here:

- Those tools do not allow us to generate a WAB file directly, although it would be reasonable to assume such a capability would be high on the priority list of future plugins.
IBM Integrated Web Application Server for i

- The .jar file generated by those tools is NOT usable. If you generate a .jar file and attempt to install it into IAS or attempt a pre-install validation of the file, it will fail.

**Note:** If application uses SPI, the MANIFEST.MF file must be manually updated with Require-Bundle (com.ibm.i5OS.lwiadmin) and Import-Package (any SPI java packages used by the application, such as com.ibm.lwi.admin.spi.server or com.ibm.lwi.admin.spi.database.

**WAR File Creation Example using RAD**

In this example of creating a war file, we will be using Rational Application Developer (RAD) 7.5. There are other tools mentioned earlier that can also be used to create these files. Once you bring up the workspace, you will have an image similar to this.

![Rational Application Developer Workspace](image)

**Figure 34 - Rational Application Developer Workspace**

The next step is to create a project. Select File-> New-> Dynamic Web Project
Figure 35 - New Dynamic Web Project

Uncheck the “Add project to an EAR” category, and select “Finish”

Figure 36 - Change perspective prompt

You will be presented to change to web perspective. Select “No” as we will want to stay in the Java EE perspective in this example.

In this example, we are just assembling the .war file. We have the pieces to do so, including the .java, .html, and .jar file. We will import these to the project. The order in which these are imported does not make a difference.

Right click on the WebContent folder, select Import, then in the source field, type in file system. This will show under the general folder the file system category. Select this and press Next.
Figure 37 - Specify Imports
Browse to the directory that the files exist.

Figure 38 - Browse to import file location
You will need to select the resources to import. We want just the index.html in this example.
IBM Integrated Web Application Server for i

Figure 39 - Specify import resources

Select “Finish” and you will see this in the WebContent section.
IBM Integrated Web Application Server for i

We now need to add the code. We have a .java file, as well as a .jar that is required to compile. This .java file is part of a package. Select the src folder, right click and select New package.

Figure 40 - Workspace view after import

Figure 41 - New Java Package prompt
After the package is created, right click on it and select import. Select file system. Navigate to the location of the .java file. Select this and press “Finish”
IBM Integrated Web Application Server for i

You will now notice the .java file imported. Also you can see there are errors shown. This example shows errors because the classes it imports are in a .jar file that is not included in the project.

![Figure 44 - Workspace view with Java errors](image)

We need to import this .jar file. These .jar files can reside in the /WebContent/WEB-INF/lib directory. Right click on the WEB-INF folder select Import. Again we will use file system and navigate to folder where the .jar file exists.
Figure 45 - Import required jar

Clicking Finish, you can see the errors now are resolved

Figure 46 - Workspace view with errors resolved
It is almost finished. There is another step required for this example. We need to identify the servlet and the servlet mapping. Under the deployment descriptor, there is a servlets category. Right click on the servlets, select New->Servlet. Since it is an existing servlet, check the “Use an existing Servlet...” box.

Select the “Browse” button, and a list of the available servlets shows up. We select the “ListPlatformAttributesServlet”, and press OK.
The next image shows the URL mapping as well. It defaults to the servlet name, so we will keep it as is.

Figure 49 - Create servlet specify URL mappings

Selecting "Finish" you will see the servlet and servlet mapping show up in the project.

Figure 50 - Workspace showing servlets and mappings
IBM Integrated Web Application Server for i

It is now ready for export. Right click on the project, select Export-> War file. On this screen, note the web project and location of the .war file. After “Finish”, the SimpleWebApplication.war file is ready for deployment to the IAS server instance.

![Export the WAR file](image)

The .war file from RAD or WDSci can be installed directly via the GUI, or programmatically using the Installing example code in Appendix D. As part of that install, the install process will run the .war file through a WAB conversion process, and a WAB jar file will be created in the same location as the .war file. Additionally, the WAB jar file can be created by executing the Pre-install Validation example code in Appendix D.

Regardless of how it was generated, the WAB-enabled .jar file is the package of choice for deploying applications into IAS.

### 2.3 Application Deployment

#### 2.3.1 Deployment via the Web Administration GUI

The Web Admin GUI is a great tool for development of the application. From an ISV perspective, the programmatic solution will be more interesting especially for delivering your application solution.

The first step is to move the .war file from RAD or WDSci, or the WAB-enabled .jar file, to a location in the IFS of the target system.
IBM Integrated Web Application Server for i

Access the GUI. Sign in and navigate to the IAS server that you want to work with.

![Figure 52 - Manage IAS server - Select server](image)

To install an application, click on the **Install New Application** from the left-hand pane or after clicking **Manage Installed Applications**, click on the **Install** button under the table of installed applications.

![Figure 53 - Manage Installed Applications](image)
IBM Integrated Web Application Server for i

The Install New Application wizard will launch. The first panel is where you can specify the location of the application in the IFS. You can either enter the path to the application or you can click on the browse button to launch a browse utility for the IFS on this system. From this panel you can specify either the '.war' file or the WAB ('.jar') file for the application.

Note: The IAS serve actually requires that the application be in the WAB file format. If you specify a .war file (which is likely as that is the easiest format to generate using the IDE tool such as RDI or WDSC) the wizard takes care of converting the .war file into the necessary WAB format.

Navigate the browse window to the location of your application. Click on the .war file for the application. Click the OK button to close the browse utility. Once you have selected an application and click ok, the path field in the Install New Application wizard will be filled in.
Click on the Next button on the Path panel of the Install New Application wizard. The wizard at this point will convert the selected '.war' file into the necessary WAB file. The application is converted, and a level of verification is done on the WAB file to help ensure that is correct. The converted WAB file version of the application is placed in the same IFS path that the '.war' file version is located.

**Note:** This is the name that will show up in the /www/<IAS server name>/liwi/apps/eclipse/plugins path (the installed applications path). This is also the name that will show up in the Manifest.mf file for the application, as the Bundle-SymbolicName property.

**Note:** In the Manifest.mf file, that a Bundle-Name property has been added, with the name of the WAB file that was installed as its value.

**Note:** If the application you are installing has a large number of JSPs contained within the .war file, the process of creating the WAB will need to compile all of these JSP. This can take a few moments to complete for very large numbers of JSPs. The alternative is to use the programmable interface to convert the .war file to the WAB format. This will allow the processing to be done outside of the GUI wizard interface.
The next panel of the Install New Application wizard is where you can specify the name and context root for this application.

The default name for the application is the name of the ‘.war’ or WAB file. This value can be changed to a value that can better identify this application. This will be the name that will be included in the Manage Applications list.

The context root is the path that is used in a client request to access the application resources. The default value is harvested from the application as part of the application verification done by the wizard.

- For example, in the request: http://myhost:80/myApplication/myJSP.jsp, “/myApplication” is the context root. Once again, the default is the name of the WAR or WAB file, appended to a forward slash (/).
Specify the port mapping for the context root. The default is to allow the context root (the application) to be accessed on any port that is valid for the server in which the application is being installed. To limit the ports to a subset of the valid ports for the IAS server, use the Specify custom port mapping option and select the desired ports from the resulting displayed list. Select Next.

Note: After the application installation is finished, the application server needs to be restarted before the specified custom port mapping takes effect.
Verify your application install selections on the Summary page and select Finish.

The application will be installed and the GUI will divert to the home main page for the selected, listing the applications installed in the server, including the one just installed. The application just installed will likely be in the process of installing.
Figure 59 - Manage Application, Application installing

Select Manage Installed Applications to monitor the status of the application installation. **Note:** If the IAS server was active at the time the application was installed, the application will automatically start, since the default behavior for an application is autostart=true.
2.3.2 Programatically via exit/install program

Using the IAS APIs, it allows more flexibility in customizing the server creation, as well as deployment of the application. These examples both illustrate the use of sample java classes and the direct use of the APIs in creating an application server and deploying of the application.

1. The creation of an IAS instance is rather simple. The customization parameters are done in a .properties file. The example we use here is called createInstance.properties

   a. place the createInstance.properties file in a directory of the IFS /home/IASServerCode. This createInstance.properties file contains the properties defining the server

   b. edit the createInstance.properties file. An example of this file is in the java code samples. The required properties to be filled in include

      - com.ibm.lwi.instance.name=
      - com.ibm.lwi.profile=
      - com.ibm.lwi.instance.web.port=

   The supplied createInstance.properties file has comments describing each property as well as the defaults if left blank.

   c. java -cp /QIBM/ProdData/OS400/jt400/lib/jt400Native.jar:/QIBM/ProdData/OS/OSG/LWI71/native/iasadmin.jar com.ibm.lwi.admin.IntegratedServerAdmin -createServer /home/LWIJavaCode/createInstance.properties
IBM Integrated Web Application Server for i

2. This code example will allow to deploy an application to an IAS server. The code involved includes an App_installApplication.java file, run.sh, and installApplication.properties.

a. Place the run.sh, App_installApplication.java, and installApplication.properties file in the IFS. For this example, it is /home/IAS/java.
b. Edit the installApplication.properties file to the appropriate values. In this example, change the following:
   - com.ibm.lwi.instance.name=
   - com.ibm.lwi.application.path=
   - com.ibm.lwi.application.bundleSymbolicName=

c. STRQSH
d. cd /home/IAS/java
e. run.sh App_installApplication /home/IAS/java/installApplication.properties

2.3.3 Programmatically via SAVF

Moving an IAS instance from one system to another requires a SAV of the proper IFS directory to a savf, then RST on the target system, followed by registering the new server.

1. Create the server. An example of this is shown in section 2.3.2 step 1a-c
2. Deploy the application. An example is in section 2.3.2 step 2.
3. CRTSAVF QGPL/IASSAVF
4. SAV DEV(/qsys.lib/qgpl.lib/iassavf.file') OBJ('/www/lwiInstanceName') PVTAUT(*YES)
5. CRTSAVF QGPL/IASSAVF2
6. SAVOBJ OBJ(QATMHINSTC) LIB(QUSRSYS) DEV(*SAVF) OBJTYPE(*FILE) SAVF(QGPL/IASSAVF2) FILEMBR((QATMHINSTC(<Target_Server_Name>)))
7. Move the save file from the source system to target system.
8. On the target system, restore the instance directory with RST DEV(/qsys.lib/qgpl.lib/lwisavf.file') OBJ('/www/lwiInstanceName') PVTAUT(*YES)
9. Restore the member of the QATMHINSTC file. To be sure, please save the existing qursys/qatmhinstc to a save file before doing the restore. It is important to specify the following options:
   - MBROPT(*NEW) and ALWOBJDIF(*FILELVL).
10. The last step is to register the IAS server.
   a. strqsh
   b. java -cp /QIBM/ProdData/OS400/jt400/lib/jt400Native.jar:/QIBM/ProdData/OS/OS
2.3.4 Administering Applications

Starting/Stopping

Autostart

Lazystart

2.3.5 Executing Application

1. Start the server
2. Start the application (if not autostart)
3. The URL http://<target system>:<IAS Apache instance port>/<application context root>/<servlet/jsp/html name>, where:
   a. target system is the TCP/IP access to the system on which the IAS instance Apache server resides. This can be a system name (short or qualified) or an IP address; it all depends on how the system is configured in the network.
   b. IAS Apache instance port is the port that was created for the IBM Apache HTTP server when the IAS instance was created. To determine this port:
      i. TBD
      ii. TBD
      iii. ...
   c. application context root is the context root specified when the application was installed. To find this value:
      iv. TBD
      v. TBD
      vi. ...
   d. servlet/jsp/html name is that initial entry point into the application.
      Commonly, it is index.html or index.htm. However it could be any .jsp file and could even be a servlet name.

4. The application should be able to be accessed via the internal IAS port(s) in addition to the HTTP port in the step above. Simply substitute the valid application ports, specified when the application was installed, for the IAS Apache instance port. Generally, the Apache port is highly recommended for production environments; however the IAS ports can be used in other environments, and is also a good debug technique when encountering a server and/or application that can not be accessed.
3. Part III – Database

3.1 Concepts

3.1.1 General Concepts

- The properties that are supported for a DB are determined by the underlying JDBC provider (driver). See the documentation for that provider for further details.
  - Others, such as Derby Cloudscape and or native JDBC driver properties?

- The default JNDI name for lookup is jdbc/ProxyDS

- Currently, the SPI provides a capability to manage only a subset of the DB properties. Alternatively, the underlying proxy classes can be used to manage other properties.

- The database connection properties are contained in the /www/<iasSvrName>/lwi/conf/overrides/database.properties file, where <iasSvrName> is the name of the IAS server instance in whose database properties you are interested.

---

Supported Databases

<table>
<thead>
<tr>
<th>Database</th>
<th>Location</th>
<th>Service name / Provider name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>remote</td>
<td>com.ibm.lwi.database.providers.DB2i5RemoteProvider</td>
</tr>
<tr>
<td>DB2</td>
<td>local (native)</td>
<td>com.ibm.lwi.database.providers.DB2i5NativeProvider</td>
</tr>
<tr>
<td>Derby Cloudscape</td>
<td>internal</td>
<td>com.ibm.lwi.database.providers.derby</td>
</tr>
</tbody>
</table>

Figure 61 - Supported Databases table
IBM Integrated Web Application Server for i

Connection ID

One of the key underlying concepts for database use is that of the connection ID.

- A connection ID is a name that is used to uniquely identify one set of DB access criteria from another. That criteria includes, among others:
  - DB type and version (i.e. DB2, Derby Cloudscape, MS SQL, etc.)
  - Platform (system) on which the DB resides
  - JDBC driver to be used for SQL statements and stored procedures
  - Libraries to be searched for the requested tables
  - Userid and password to be used to access the DB

- The /www/<IAS server name>/lwi/conf/overrides/database.properties file contains the definitions of the connection IDs for that server. The connection ID itself becomes the prefix for the properties in the file. For instance, if you select the *DEFAULT connection id name when creating a connection (there can be only one *DEFAULT connection id per server), then the properties for that connection would look as such:
  - .libraries=TESTLIB
  - .defaultAutoCommit=false

However, if you define a connection ID of devConn, those same properties would look like this:
  - devConn.libraries=TESTLIB
  - devConn.defaultAutoCommit=false

- A connection id can not be used across IAS instances. However, all applications installed to an IAS instance can access any connection id defined for that instance.

Database Properties

The following table outlines the Connection properties that can be managed within the IAS framework.

<table>
<thead>
<tr>
<th>Property name</th>
<th>Default value / Optional values</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbprovider</td>
<td>Optional values:</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>- com.ibm.lwi.database.providers.DerbyProvider</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- com.ibm.lwi.database.providers.DB2i5RemoteProvider</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- com.ibm.lwi.database.providers.DB2i5NativeProvider</td>
<td></td>
</tr>
<tr>
<td>dbname</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>dbuser</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>dbpassword</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>dbservername</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>dbport</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>enableConnectionPooling</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>maxActive</td>
<td>8</td>
<td>no</td>
</tr>
<tr>
<td>maxIdle</td>
<td>8</td>
<td>no</td>
</tr>
<tr>
<td>Property name</td>
<td>Default value / Optional values</td>
<td>Mandatory</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>maxWait</td>
<td>-1</td>
<td>no</td>
</tr>
<tr>
<td>whenExhaustedAction</td>
<td>Default:WHEN_EXHAUSTED_GROW options: -WHEN_EXHAUSTED_FAIL -WHEN_EXHAUSTED_GROW -WHEN_EXHAUSTED_BLOCK</td>
<td>no</td>
</tr>
<tr>
<td>testOnBorrow</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>testOnReturn</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>testWhileIdle</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>timeBetweenEvictionRunsMillis</td>
<td>-1</td>
<td>no</td>
</tr>
<tr>
<td>minEvictableIdleTimeMillis</td>
<td>1000 * 60 * 30</td>
<td>no</td>
</tr>
<tr>
<td>validationQuery</td>
<td>null</td>
<td>no</td>
</tr>
<tr>
<td>defaultReadOnly</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>defaultAutoCommit</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>defaultTransactionIsolation</td>
<td>Default: UNKNOWN_TRANSACTIONISOLATION options: -UNKNOWN_TRANSACTIONISOLATION - TRANSACTION_READ_UNCOMMITTED -TRANSACTION_READ_COMMITTED -TRANSACTION_REPEATABLE_READ -TRANSACTION_SERIALIZABLE</td>
<td>no</td>
</tr>
</tbody>
</table>

Figure 62 - Connection ID Properties (database.properties file)

Default Properties File example

Following is an example of database.properties file entries for a default connection id that access a remote DB:

```java
.defaultAutoCommit=false
.rollbackOnClean=false
.maxActive=8
.defaultTransactionIsolation=2
.dbprovider=com.ibm.lwi.database.providers.DB2i5RemoteProvider
.poolPreparedStatements=false
dbpassword = {aes:3C5SnKQL63SjkEy44Gs+vHF6nQzC+Dil1NzNvSiAzzk=}UJOS5etoVSMMs9S2pz807Q==
dbuser=macrafic
.libraries=TESTLIB
.maxOpenPreparedStatements=0
dbservername=DEMOPROD
```

Adding Another DB connection (local) called LclDBCon:

The properties are added to the same database.properties file as above and are the same as the above, with two exceptions:
IBM Integrated Web Application Server for i

- The connection id (LclDBCon) precedes each property
- The DB provider is different

**Example:**
LclDBCon.defaultAutoCommit=false
LclDBCon.rollbackOnClean=false
...
LclDBCon.dbprovider=com.ibm.lwi.database.providers.DB2i5NativeProvider

---

**Getting an Encrypted Password**

If a database configuration needs a password, use the following command to get an encrypted password: /www/<ias instance name>/lwi/bin/lwencoder

    lwencoder -filename ..\conf\database.properties –keylist <conn ID>.dbpassword

Note, however, that when creating a DB Connection, either via the GUI or programmatically, the desired password can be entered “in the clear” and will automatically be encrypted by the underlying Connection create logic prior to it being placed into the database.properties file.

**Library Lists**

- Multiple libraries are specified in the /www/<iasSvrName>/lwi/conf/overrides/database.properties file as follows:
  - <connID>.libraries=*LIBL
  - <connID>.libraries=*LIBL Lib1 Lib2
  - <connID>.libraries=Lib1 Lib2

  where *LIBL is a placeholder for the current library list of the server job.

- *LIBL general rule:
  - When the first entry is *LIBL, the specified libraries are added to the current library list of the server job
  - When you do not use *LIBL, the specified libraries replace the current library list of the server job

- Beyond the *LIBL rule, the role of the naming property determines which library(ies) is/are searched when an SQL statement is not library-qualified. SQL naming (sql) is the default (vs system). The library list table at [http://publib.boulder.ibm.com/infocenter/iseseries/v5r4/index.jsp?topic=/rzh/hh/jdbcliblistproperty.htm](http://publib.boulder.ibm.com/infocenter/iseseries/v5r4/index.jsp?topic=/rzh/hh/jdbcliblistproperty.htm) explains the differences between sql and sys-
IBM Integrated Web Application Server for i

"tem naming. It is important to understand these premises, or the wrong data will be targeted.

Creating DB Connections via the GUI

This section is here just for reference. As stated in the Introduction, this document is targeted to the ISV and, thus, the more programmatic solutions to these scenarios.

Access the GUI. Sign in and navigate to the IAS server that you want to work with.

![Figure 63 - Web Admin GUI Manage Database Connections](image)

Click on the Manage Database Connections link in the left navigation area. When you first enter the Manage Database Connections interface the table will likely be empty. There are not database connections created as part of the server creation process.
Click on the Create Database Connections link in the left navigation area, or click on the Create button under the database connections table to launch the Create Database Connection wizard.

**Figure 65 - Create Database Connections wizard – Intro**
IBM Integrated Web Application Server for i

The Create Database Connection wizard walks you through all the necessary steps to create the database connection. Help you figure out what database connection type to create, where the database resides, and the user profile needed to access that database,

The first page of the wizard, you can specify the type of database connection protocol and what type of database you want to use.

The database options supported on IBM i:
- DB2
- Derby

There are 3 different database connection types currently supported on IBM i:
- IBM Developer Kit for Java JDBC™ driver - Local DB2
- IBM Toolkit for Java JDBC™ driver - Remote DB2
- Derby Database Provider – Local

Connection ID – distinguishes this connection from others. This is the connection ID that will be used in the application code when it references a database. If only a single database is required, you can leave the default values of ‘DEFAULT’. This will generate a generic ID, that will allow you in the application code to reference the database in a generic manner. If you require multiple databases for this application, you must specify unique connection IDs that your application code will need to use.

Once you have specified the database connection type and the connection ID, click Next.
The database connection information page is where you actually specify the location of the database and the user profile that will be used when accessing the database.

**Database location:** The location for this database. For a database that resides on this same IBM i, the special value of ":LOCAL" should be used. If the database resides on a different IBM i system, you would want to specify the address to the remote system.

**Schema name:** The name of the database. Specify the Schema, collection or library name of the database.

**Connection user ID:** The user ID that will be used by the application server when connecting to the database to access or write data.

**Password:** The password that will be used with the Connection user ID value.
Create Database Connection

JNDI Name

The Java Naming and Directory Interface™ (JNDI) provides naming and directory functionality to Java applications. An application uses a JNDI name along with a database connection to access a database. The Integrated Web application server provides one JNDI name to be used by any application that needs database access. Access to different databases is achieved by using the same JNDI name with different database connections. Database connections are distinguished from each other by connection ID's.

The following JNDI name must be used by applications to access databases:

JNDI name: jdbc/ProxyDS

Figure 67 - Create Database Connection - JNDI name

The Java naming and Directory Interface™ (JNDI) provides naming and directory functionality to the Java application. The application uses the Connection ID along with the JNDI name to identify and access database content. Currently IAS has a limitation in regard to the JNDI value. The JNDI value must be a predetermined value. There is currently no ability to specify a unique value. The JNDI name must be jdbc/ProxyDS.

Click Next to continue.
The Summary page shows all the values that have been specified thru the wizard. Click Finish to complete the creation of the database connection.

Figure 69 - Manage Database Connections

- 69 -
Managing Connections Programmatically

Note: Not all of the properties can be managed programmatically using the SPI. Reference the JavaDoc for the SPI to determine which properties can be managed this way. For the rest of the properties, one of three different techniques would have to be used:

- Manually updating the database.properties file
- Writing a program that updates the database.properties file directly
- Using the proxy support that actually underlies (backs up) the provided SPI interfaces.

The proxy support is very similar to the SPI interface. In fact, only connection configuration differs; using the connection to execute SQL is the same, regardless of the method chosen to configure the connection. The difference between the two approaches is demonstrated in these two code fragments:

**SPI support (configuring the connection and then using the connection):**

```java
ServerManager sm = ServerManagerFactory.getServerManager();
ManageableInstance ins = sm.getServer(<IAS_server_name>);
DatabaseManager dbMgr = ins.getDatabaseManager();
Connection connID = (Connection)
   dbMgr.getDatabaseConnection(<connection_id>);
connID.setDefaultAutoCommit(true);
connID.setPoolPreparedStatements(true);
connID.setMaxOpenPreparedStatements (100);
...
InitialContext ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/ProxyDS");
Connection dbConnect = ds.getConnection(<connection_id>,
   null);
String sqlText = <some valid SQL query statement text>;
PreparedStatement stmt = dbConnect.prepareStatement(sqlText);
ResultSet rs = stmt.executeQuery();
...
```

**Proxy support (configuring the connection and then using the connection):**

```java
InitialContext ctx = new InitialContext();
IProxyDataSource ds = (IProxyDataSource)
   ctx.lookup("jdbc/ProxyDS");
IDatabaseConfig config =
   ds.getDatabaseConfig(<connection_id>);
   config.setMaxActive(5);
   config.setWhenExhaustedAction(IDatabaseConfig.WHEN_EXHAUSTED_GROW);
...
```
IBM Integrated Web Application Server for i

    Connection dbConnect =
    ds.getConnection(<connection_id>);
    String sqlText = <some valid SQL query statement text>;
    PreparedStatement stmt = dbConnect.prepareStatement(sqlText);
    ResultSet rs = stmt.executeQuery();

**Note:** To be able to access the proxy classes in your RAD workspace, the following .jar file must be added to your project:

```
com.ibm.lwi.database.datasources.proxy_<version>-LWI.jar
```

in

```
<server_path>/lwi/runtime/database/eclipse/plugins
```

where `<server_path>` is a path to an existing IAS server (any IAS server will do). It is suggested that you import this jar file from the server into a location of your choice on your workstation, then change the Java Build Path to include the jars from your workstation. To Change the Java Build Path, right click your project in the Package Explorer and select the context menu item, Properties. Then select Java Build Path from the left hand side of the ensuing pop-up window, select the Libraries path, and select Add External Jars and point to your jars.

Nothing special should be needed on your server at runtime. The Application install logic adds the correct package imports to the application’s manifest to bring the appropriate JAR files into the environment. In addition, the profiles supported on System I always include the ‘database’ profile, which ensures the proxy classes/jars are available at runtime.

**Pooling**

By default, DB connection pooling is enabled, but Prepared Statement caching is not. For performance reasons, it is suggested that both of these pooling capabilities be enabled.

DB connection pooling allows a pool of connections to be established to the DB and subsequently re-used. Since establishing a connection is one of the top two performance consumers, being able to re-use the connections is essential. Also, the ability to re-use a connection means that fewer connections can be used to satisfy the peak demands of the application. Fewer connections mean fewer system memory resources are required.

It is also strongly recommended that PreparedStatements be used in an application wherever they can. In fact, while there are reasons for not using PreparedStatements, any variance from this practice should be reviewed carefully. Similar to the behavior of the DB connections themselves, preparing an SQL statement for execution is a large part of the performance consumption of executing the SQL. When the preparation can be cached and re-used, significant performance benefits can be achieved. That is the purpose of the Prepared Statement cache.

**Connections**

As already stated, pooling of DB connections is enabled by default. The database.properties file will not have an entry indicating this, since it is the default. If
IBM Integrated Web Application Server for i

If an entry were present, you would see, `<connectionID>.enableConnectionPooling=true`. As might be expected, providing a value of `false` disables connection pooling, irregardless of any other properties that exist in the file pertaining to DB connection pooling.

Other properties that impact DB connection pooling include:

- `maxActive` - Maximum number of objects that can be borrowed from the pool at one time (concurrently). When non-positive, there is no limit to the number of objects that may be concurrently active. When this number is exceeded, the pool is said to be exhausted. The default is 8.
- `maxIdle` - Objects that can sit idle in the pool at any time. When negative, there is no limit to the number of objects that may be idle at one time. The default is 8. Normally, this value would be the same as the `maxActive` value.
- `maxWait` - The maximum amount of time to wait for an idle object when the pool is exhausted and `whenExhaustedAction` is `WHEN_EXHAUSTED_BLOCK`. The default is -1.
- `whenExhaustedAction`
- `testOnBorrow`
- `testOnReturn`
- `testWhileIdle`
- `timeBetweenEvictionRunsMillis`
- `minEvictableIdleTimeMillis`
- `validationQuery`

Note: Not all of the above properties are able to be managed programmatically via the SPI interfaces provided. However, they can be accessed via the underlying proxy support.

Prepared Statements

Prepared Statement caching (pooling) is disabled by default. The database.properties file will have an entry as such: `<connectionID>.poolPreparedStatements=false`. As might be expected, providing a value of `true` will enable the caching capability.

There is only another property that can be managed for the cache:

- `maxOpenPreparedStatements` – The maximum number of Prepared Statements that the cache can contain. This number should be large enough to accommodate all the Prepared Statements used by all the applications (installed in the same server) which leverage this connection id. The default is 100.

Updating Using the GUI

The Web Performance Advisor within the Web Admin GUI interface provides the interface necessary to update the prepared statement and pooling values. See the Web Performance Advisor section for details on access WPA. Once you have accessed WPA, select the IAS server and click ok the Manage Attributes
IBM Integrated Web Application Server for i

button. The *Database Connections* tab is the starting point for updating and database property values.

The *Database Connections* tab shows all the database connections that have been created for this application server. To work with the performance attributes for a specific database connection, click on the *Database Connection ID* value. This is a link to the Manage database connection attributes page.

Figure 70 - Web Performance Advisor - Database Connections tab

Update the database pooling options. Click on the *Advertise* link to get an explanation for each attribute along with the recommended value. Enable database pooling along with the correct connection pooling values.

Figure 71 - Web Performance Advisor - Manage Database Connection Attributes
IBM Integrated Web Application Server for i

Languages

To specify what language the server should use to perform stores and perform sorts, use the following properties:
<connectionID>.sort=language
<connectionID>.sortLanguage=ita

For a list of supported languages, see http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/nls/rbagsingidsdefaultccsids.htm.

Miscellaneous

For the Database capabilities of IAS to execute properly, the QCCSID system value on the server can NOT be 65535. It must be set to some other valid value (most commonly, in the U.S., to 37). Use the wrksysval qccsid 5250 command, and option 5 (Display) to view the current value on the system. The same command, option 2 (Change) will allow you to change the value. For a list of supported CCSID values and their definitions, see http://www-03.ibm.com/systems/i/software/globalization/ccsid_list.html, or http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/nls/rbgsccsidcdepgscharsets.htm.
4. Part IV – Miscellaneous

4.1 General Debug Concepts

4.1.1 Server

4.1.2 Application

4.1.3 Admin Servers

Types of Admin Servers

Directory Structure

4.1.4 Logging/Tracing

This is not intended to be a discourse on logging and tracing; some knowledge of common logging and tracing technology, particularly the JSR47 (java.util.logging) logging, is assumed. This section is included to provide a practical jump-start to implementing logging / tracing into an application.

The underlying IAS logging system supports several different logging interfaces, including:

- JSR47 (java.util.logging)
- IBM Commons Logging (ICL)
- Apache Jakarta Commons
- Eclipse Logging
- OSGi Logging

In addition, the SPI for IAS provides a layer on top of the JSR47 technology. This section will focus primarily on using the SPI and JSR47 interfaces. Some minimal information is also provided on leveraging ICL.

Logging Interfaces

**IBM Commons Logging (ICL)**

As previously indicated, this section will focus on JSR47 and on the IAS SPI logging interface. However, for those wanting to use ICL, that support is now provided in a separate plug-in, com.ibm.logging.icl. Deployers wanting to use ICL will need to add to an application’s /www/IAS instance name/IWI/apps/eclipse/plugins/<application name>/META-INF/MANIFEST.MF file the following statement in order to import the needed ICL packages or logging and tracing: *Import-Package: com.ibm.logging.icl.*
IAS Logging Interface

Generally, the JSR47 logging interface would be used by the deployer for their application. However, if the deployer wants to extend the IAS SPI, particularly if they want to add their own resource bundle(s) for internationalizing messages, then the IAS provides an interface to the JSR47 logging that makes this easier. This interface is provided by 2 classes and a property file:

- class LogWriter, which represents a java.util.logging.Logger instance
- class MessageWrapper, which provides the capability to wrap messages from a message resource bundle and appropriate message data together into a single entity for logging.
- user-provided .properties file containing the application messages (message resource bundle)

To implement this type of logging:

5. Build the message resource bundle. This would have entries similar to the following. These entries would subsequently be used to output all messages (screen messages, logging/tracing messages, etc.) in the application.
   - IAS00201E=IAS00201E – File or directory path {0} already exists.
   - IAS00202E=IAS00202E – Value {0} for property {1} is not valid.
   See the com.ibm.lwi.admin.spi.resources.Messages.properties file for more detailed examples.

6. Register the message resource bundle (.properties file) with the static LogWriter method registerMessageResourceBundle(String bundleName). This determines where the LogWriter looks for the message associated with the message id provided when a log request is made.

7. Use code similar to the following 2 examples to log messages retrieved from your message resource bundle (These examples use the example messages defined above, to demonstrate substitution of message data into the message):
   ```java
   MessageWrapper pathInvalid = new MessageWrapper("IAS00201E", dirPath);
   LogWriter.severe(pathInvalid);
   
   MessageWrapper propInvalid = new MessageWrapper("IAS00202E ",
                                                    new String[] { propValue, propName});
   LogWriter.warning(propInvalid);
   ```

JSR47 Logging

JSR47 is the core of all logging provided by IAS, regardless of which interface is used. This is the logging interface recommended to deployers for their application. To obtain a logger for use within an application, code similar to the following would be used:

```java
Logger logger = Logger.getLogger(<package name>);
logger.setParent(Logger.getLogger(""));
logger.setUseParentHandlers(true);
logger.setLevel(Level.FINEST);
```
where *<package name>* is the name of the Java package for which logging is desired. All classes within this *<package name>* and all classes within sub-packages are eligible to be logged.

Generally, the *<package name>* used in the code will be the “lowest common denominator” package that encompasses all of the application. That way, logging can be done throughout the entire application and there is no responsibility at development time, to determine the areas of the application that should be logged or the amount/type of logging done. Once the logger is obtained, any of the java.util.logging.Logger methods (entering, exiting, log, logp, info, warning, fine, etc..) may be used in the application, where and as appropriate.

At runtime, the logging.properties file determines the amount, type and target areas of the application that should be logged to the file(s). Statements similar to the following:

```
com.mycompany.myapp.level=FINEST
```

would be added to determine the level of logging to be done on the application and the packages to be logged. As explained elsewhere in this section, the file can be changed, and the lwilog command run to dynamically change the logging without having to recycle the IAS instance.

Typically, a configuration similar to the default logging configuration is used in production. This basically provides exception/error logic without excessive logging (excessive logging negatively impacts an application’s performance). Only when specific issues require additional information to be logged is the base configuration changed.

**Logging Architecture**

All the logging interfaces supported by IAS eventually have their messages processed by a JSR47 logger. The actual flow of messages issued by applications using the different logging interfaces is shown in the diagram below:
Log and Trace Event Flow

Client App1 \(\rightarrow\) Client App2 \(\rightarrow\) Client App3 \(\rightarrow\) Client App4 \(\rightarrow\) Client App5

- Eclipse Logging
- OSGi Logging
- WED LogRedirector
- IBM Commons Logging
- Jakarta Commons

JSR47 Logger

CBE Generator (JSR47 Handler)

Log File \(\rightarrow\) Trace File

Figure 72 - Logging and Tracing Architecture

Logging/Tracing Levels

The following table defines the various levels of logging/tracing that can be lever-aged.

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Level</th>
<th>Content / Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td></td>
<td>No events are logged or traced</td>
</tr>
<tr>
<td>Logging</td>
<td>Severe</td>
<td>Serious failure. Events that are of considerable importance and which will prevent normal program execution. Should be reasonably intelligible to end users and system administrators.</td>
</tr>
<tr>
<td></td>
<td>Warning</td>
<td>Potential error or impending error. Events that will be of interest to end users or system managers, or which indicate potential problems.</td>
</tr>
<tr>
<td></td>
<td>Info</td>
<td>General information outlining overall task progress. Typically will be written to the console or its equivalent. Only used for reasonably significant messages that will make sense to end users and system admins.</td>
</tr>
</tbody>
</table>
## IBM Integrated Web Application Server for i

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Level</th>
<th>Content / Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config</td>
<td></td>
<td>Configuration change or status. Intended to provide a variety of static configuration information, to assist in debugging problems that may be associated with particular configurations. For example, CONFIG message might include the CPU type, the graphics depth, the GUI look-and-feel, etc.</td>
</tr>
<tr>
<td>Fine</td>
<td></td>
<td>Trace information – General trace + method entry / exit / return values. Should be used for information that will be broadly interesting to developers who do not have a specialized interest in the specific subsystem. FINE messages might include things like minor (recoverable) failures. Issues indicating potential performance problems are also worth logging as FINE.</td>
</tr>
<tr>
<td>Finer</td>
<td></td>
<td>Tracing information – Detailed trace. Indicates a fairly detailed tracing message. By default, logging calls for entering, returning, or throwing an exception are traced at this level.</td>
</tr>
<tr>
<td>Finest</td>
<td></td>
<td>Tracing information – A more detailed trace – Includes all the detail that is needed to debug problems.</td>
</tr>
<tr>
<td>All</td>
<td></td>
<td>All events are logged. If you create custom levels, All includes your custom levels, and can provide a more detailed trace than Finest.</td>
</tr>
</tbody>
</table>

**Figure 73 - Logging and Tracing Levels**

The IAS logging system is more clearly defined into two separate entities, **logging** and **tracing**. These two entities have the following characteristics:

- **Logging**
  - **Logging** is considered on for all components (both IAS and deployer provided), and is used for standard problem determination of components.
  - **Logging** messages are considered user consumable. These messages are meant for consumption by system administrators, web administrators, end users, etc.
  - **Logging** messages require localization (they are translatable).
  - **Logging** messages have a severity level of INFO, WARNING, or SEVERE from a JSR47 perspective.
  - **Logging** messages have an associated message id.
  - **Logging** messages should be issued when any exception is caught, or in other unexpected error conditions.
  - **Logging** messages are written to a separate configurable log file. The default logging file is located at `<lwi_install_directory>/logs/error-log-X.xml/html`.  

- **Tracing**
  - **Tracing** is not on for all components (both IAS and deployer provided) by default. Tracing is used for advanced diagnostics and problem determination.
  - **Tracing** messages are meant for consumption by IBM Field Support and developers/testers.
  - **Tracing** messages do not require localization.
  - **Tracing** messages have a severity level of CONFIG, FINE, FINER, FINEST from a JSR47 perspective.
  - **Tracing** messages do not require an associated message id.
IBM Integrated Web Application Server for i

- **Tracing** messages should provide a meaningful trace through a component. Minimally, there should be messages issued when entering and exiting methods. In addition, messages should be issued in complex code, or anywhere additional tracing would be deemed helpful to IBM Field Support.

- **Tracing** messages are written to a separate configurable log file. The default tracing file is located at `<lwi_install_directory>/logs/trace-log-X.xml/html`.

The summary table below shows the mapping of message levels to log/trace files for the default configuration (logging.properties file) shipped with IAS. In general, the "rules of thumb" are:

- `lwistdout.txt` shows information generated from the trace and debug methods
- `lwistderr.txt` shows information generated from the info, warning and severe methods
- `trace-log-n.xml` shows everything (the RCPTraceHandler level in the default configuration is FINEST)
- `error-log-n.xml` shows information generated from the warning and severe methods (the RCPLogHandler level in the default configuration is WARNING)
- `console` shows everything (the ConsoleHandler level in the default configuration is FINEST)
- System.out and System.err default to level INFO. As such, they will not appear in the error-log-n.xml file.
<table>
<thead>
<tr>
<th>Logging Level</th>
<th>Logging Method</th>
<th>Standalone (Console)</th>
<th>trace-log-n.xml</th>
<th>lwistdout.txt</th>
<th>lwistderr.txt</th>
<th>error-log-n.xml</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>Any method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>info</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>warning</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>severe</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>WARNING</td>
<td>warning</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>severe</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEVERE</td>
<td>severe</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONFIG</td>
<td>info</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>warning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>severe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>FINE</td>
<td>info</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>warning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>severe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>trace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINER</td>
<td>info</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>warning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>severe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>trace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINEST</td>
<td>info</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>warning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>severe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>trace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>debug</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td>info</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>warning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>severe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>trace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>debug</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System.out</td>
<td>N/A</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System.err</td>
<td>N/A</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 74 - Logging/Tracing File Mappings*
IBM Integrated Web Application Server for i

Log Formats

The **logging** and **tracing** log files generated by the IAS logging system can be produced in one of two formats, configurable in the logging.properties file:

- Common Base Event (CBE) format
- RCP format

Changing the format on the RCPTraceHandler changes the format for the trace-log-n.xml file, while changing the format on the RCPLogHandler changes the format for the error-log-n.xml file.
IBM Integrated Web Application Server for i

**CBE format** is the default format used by the IAS logging system. It produces log files with an `.xml` suffix, such as `trace-log-0.xml`. The use of XML tags in the log file reduces the ability to interpret the contents of the log file with a text editor. However, the logging system leverages XSL Transform which processes the CBE format and generates HTML. This allows CBE formatted log files to be viewed in a standard browser in an easily read format. It provides a Viewer Preferences capability which allows filtering and sorting rules to be specified for the log entries displayed in the browser table. An example of the log file displayed in a browser is shown below:

![Trace File Viewer](image)

**Figure 75 - Trace logs view**
IBM Integrated Web Application Server for i

**RCP format** produces files in a typical text format, with standard text-editor formatting (end of line characters, tabs, spacing, etc.). An example:

```
2005/12/15 17:05:19.738 SEVERE LWIN00900E - Exception occurred: This is WARNING level / severe text. :class.method
2005/12/15 17:05:19.738 SEVERE LWIN00900E - Exception occurred: This is SEVERE level / severe text. :class.method
2005/12/15 17:05:19.743 INFO LWIN00900E - Exception occurred: This is CONFIG level / severe text. :class.method
2005/12/15 17:05:19.760 WARNING LWIN00900E - Exception occurred: This is WARNING level / warning text. :class.method
2005/12/15 17:05:19.762 SEVERE LWIN00900E - Exception occurred: This is CONFIG level / severe text. :class.method
2005/12/15 17:05:19.763 INFO LWIN00900E - Exception occurred: This is FINE level / info text. :class.method
2005/12/15 17:05:19.767 WARNING LWIN00900E - Exception occurred: This is FINE level / warning text. :class.method
2005/12/15 17:05:19.769 SEVERE LWIN00900E - Exception occurred: This is FINE level / severe text. :class.method
2005/12/15 17:05:19.772 INFO LWIN00900E - Exception occurred: This is FINEST level / info text. :class.method
2005/12/15 17:05:19.774 WARNING LWIN00900E - Exception occurred: This is FINEST level / warning text. :class.method
2005/12/15 17:05:19.776 SEVERE LWIN00900E - Exception occurred: This is FINEST level / severe text. :class.method
2005/12/15 17:05:19.779 INFO LWIN00900E - Exception occurred: This is FINEST level / info text. :class.method
2005/12/15 17:05:19.781 WARNING LWIN00900E - Exception occurred: This is FINEST level / warning text. :class.method
2005/12/15 17:05:19.783 SEVERE LWIN00900E - Exception occurred: This is FINEST level / severe text. :class.method
2005/12/15 17:05:19.787 INFO LWIN00900E - Exception occurred: This is ALL level / info text. :class.method
2005/12/15 17:05:19.789 WARNING LWIN00900E - Exception occurred: This is ALL level / warning text. :class.method
2005/12/15 17:05:19.791 SEVERE LWIN00900E - Exception occurred: This is ALL level / severe text. :class.method

```

Figure 76 - RCP view

**Changing the Logging Configuration**

Logging configuration properties are consolidated in one properties file in IAS. This file is: `/www/<IAS instance name>/lwi/conf/overrides/logging.properties`. This is the typical place for deployers to make logging configuration changes (do not make changes to the logging.properties file in the `conf` subdirectory).

Should it become necessary, after making changes to the file, to revert to the default configuration again, simply copy the logging.properties file from the `/www/<IAS instance name>/lwi/conf` directory into the `overrides` subdirectory, replacing the current file.

If a deployer wishes to place the logging properties file elsewhere on their system, make the following changes:

- Create the logging properties file, using a file name of choice, and place it in the desired place in the IFS.
- Create a properties file, using a file name of choice, in the `/www/<IAS instance name>/lwi/conf/overrides/` subdirectory.
- In the newly-created properties file, add a `com.ibm.rcp.core.logger.boot.config.file` property to point to the deployer’s logging properties file. This will override the default setting in `/www/<IAS instance name>/lwi/conf/overrides/config.properties`.

By taking these steps, a deployer/user prevents IAS from changing their logging configuration properties in the case where IAS has to update the `logging.properties` file it ships.

To add an entry to the configuration file for a deployer application, something like the following would be done:

- Add a line into the `logging.properties` file to set the level of logging for the application. This is done at a package level and can be as granular as desired.
IBM Integrated Web Application Server for i

- Example – an application has several packages that start with com.mycompany.myapp. To enable the highest level of tracing for all packages that start this way, place the following line into the logging.properties file:
  com.mycompany.myapp.level=FINEST
- Example - to enable INFO level of logging for only the package com.mycompany.myapp.mysubsystem, place the following line into the logging.properties file:
  com.mycompany.myapp.mysubsystem=INFO
  - Place the desired handler(s) at the same (or lower) level as your package levels. As an example, the default level for the RCPLogHandler is WARNING. In the 2nd example above, logging for package com.mycompany.myapp.mysubsystem would not log to the RCPLogHandler, since the INFO level of the package is lower than the WARNING level of the handler. It would be necessary to change the level of the handler to INFO.

**Dynamically Changing the Logging Configuration**

IAS deployers have requested the ability to modify the log configuration properties (/www/<IAS instance name>/lwi/conf/overrides/logging.properties) and have the modified properties applied to the IAS logging system without having to restart the IAS runtime. This function is especially needed when it is necessary to obtain additional diagnostic information, particularly as a problem is occurring and the appropriate logging configuration is not in place. This is often the case in production systems, where logging levels are not generally set to produce trace/debug level information.

To provide this capability, the command, lwilog is used, which refreshes the logging configuration properties. This command should be used when the IAS runtime is active. A typical use case of how to refresh the logging configuration properties would be:
  - Change the logging.properties file as appropriate to the requirements
  - Start QShell (qsh from a 5250 command line) and change the current directory to the bin directory of the active IAS instance (cd /www/<IAS instance name>/lwi/bin).
  - Execute the lwilog command (lwilog.sh –refresh)
  - When complete, the QShell response should look similar to the following:
    refresh is done!
    SUCCESS
    $

For more details on the lwilog command, including how to add new loggers and/or handlers, see Appendix C.

**System.out and System.err**

For debugging purposes during development, developers often use the System.err and System.out output streams to issue messages. The location of where these messages are issued is controlled by two platform launch options that
IBM Integrated Web Application Server for i

would be specified in the /www/<IAS Instance name>/lwi/conf/overrides/config.properties file. By default these platform launch options are not set but, for reference, appear as comments in the /www/<IAS Instance name>/lwi/conf/config.properties file. The usage of these two platform launch options is explained in the table below:
IBM Integrated Web Application Server for i

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-separateSysLogFiles</td>
<td>If this platform launch option is set (to true) the System.out and System.err messages will not only be written to the Tracing log file, but are also be written to separate files in the &lt;lwi_install_directory&gt;/logs directory. These files are called SystemOut.log and SystemErr.log.</td>
<td>Unset by default. System.err and System.out are written to the Tracing log file, and not to separate files.</td>
</tr>
<tr>
<td>-noSysOutRedirect</td>
<td>If this platform launch line option is set, messages written to the System.out and System.err output streams are not redirected to a JSR47 logger, and therefore are not logged to any persistent file. These messages only appear in the OSGi command prompt.</td>
<td>Unset by default. System.out and System.err are redirected to the JSR47 Logger and are written to a persistent log file. The specific log file the messages are written to is based on the setting of the –separateSysLogFiles command line option.</td>
</tr>
</tbody>
</table>

Figure 77 - System.out and System.err Configuration

Multiple Environments

Every programming shop, ISV or end user, finds a need to have multiple environments, such as development, testing, performance, production, etc.. Within any particular environment, it may be necessary to be able to access different applications and different “sets” of data. The following describes some potential scenarios, and how you might address them.

The following table defines those scenarios that might be of interest. They are listed in order of increasing flexibility and productivity.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Same Server</th>
<th>Same Application</th>
<th>Same Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>4</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
Scenario 1 – Data Access
Two programmers/organizations (entities) are executing on the same server, using the same application, and accessing the same data set.

Recommended Data Access Solution
This is one of the easier scenarios; one might even say it is uninteresting. One server is defined, one DB connection id (pool) is defined, and a single application is deployed to the server. There are no special configuration actions to be taken in this scenario; both entities test/execute as usual. The application must protect against DB conflicts, but this is more an issue of good programming practice and application integrity and has little to do with scenarios as we are defining them here.

The disadvantage of this scenario is that both entities have to coordinate environment changes (recycling the server, updating the application, altering the DB connection pool). Thus, it would not necessarily be an appropriate scenario for multiple developers or development organizations working on different parts of the same application code.

Scenario 2 – Data Access
Two programmers/organizations (entities) are executing on the same server, using the same application, but with a requirement to access different data sets.

Recommended Data Access Solution
In this scenario, a single IAS server is configured, a single application is deployed, and multiple DB connection ids (pools) are defined for the server. The requirement exists for the application to access a different DB connection pool, depending on which entity is executing.

Without some assistance from the application itself, this is an impossible scenario to solve. The application would have to have logic to determine which DB connection id (pool) to use, based on the executing entity.

For applications which require this behavior to satisfy application requirements, this scenario becomes another uninteresting one. The necessary behavior is already built into the application. This might be the case, for example, in an application that hosts different companies, each requiring their own set of DB tables. The application might read in a properties file, XML file, or DB table to determine which DB connection id (pool) to use based on the executing entity/role.

The more interesting case is where the requirement to access different data sets does not emanate from the application requirements itself, but from external execution/testing requirements. Such might be the case where two developers are testing the same application on the same server, but have different sets of test cases to test, and the need to have different data to replicate those test cases. In that case, it is usually impractical to enhance the application to accommodate
IBM Integrated Web Application Server for i

this requirement, since it is a requirement of testing and not one of production. One of the other scenarios would be more appropriate.

The net on this scenario is that it is of limited use at best. In most cases, creating separate IAS instances is a better choice here.

**Scenario 3 – Data Access**

Two programmers/organizations (entities) are executing on the same server, executing different applications, but using the same data set against which to test.

**Recommended Data Access Solution**

In this scenario, a single IAS server is configured, multiple applications are deployed, and a single DB connection id (pool) is defined for the server. Since DB connection ids (pools) are shared at the server level, different applications can use the same one.

This is an uninteresting scenario. It is business as usual, no special configuration must be done, and multiple entities should not have an issue executing in this environment.

The disadvantage to this scenario is that the different entities must coordinate should it become necessary to recycle the server, reconfigure the server, or reconfigure the DB connection id (pool). They are isolated, however, at the application level, since different applications can be installed and uninstalled without impacting other applications in the server, without impacting any DB connection ids (pools) in the server and without requiring the server itself to be recycled.

If at all possible, we would recommend that the limitation to the same DB connection id (pool) be relieved, at least in any environment short of production. Multiple DB connection ids (pools) can be defined, even if they are exactly the same configuration. They simply require different ids (names).

**Scenario 4 – Data Access**

Two programmers/organizations (entities) are executing on the same server, but executing different applications and accessing different data sets.

**Recommended Data Access Solution**

This is another uninteresting scenario; in fact, it is business as usual. Most production environments execute this way. Multiple applications are deployed to a single server and each application may, or may not, access the same data sets. Again, it is the responsibility of the applications to maintain database integrity, if multiple applications are accessing the same data.

**Scenario 5 – Data Access**

Two programmers/organizations (entities) are executing on different IAS instances.

**Recommended Data Access Solution**

Since applications and DB connection ids (pools) are not shared across IAS instances, there is no potential for conflicts between the entities here, with one caveat. If the two entities are accessing the same data, the application(s) they are using must have responsibility for data integrity. This, as already having been
stated, is a requirement of good programming practice and application integrity, and has little to do with any particular execution scenario.

This is the most flexible of all the scenarios and the recommended way to do business, if at all possible.

**Overall Recommendations**

The most complex scenarios are those that apply to the development environment, since this involves several members working on the same application and potentially needing access to different data sets for testing. For this environment, the ideal situation would be one where each developer had their own IAS server to work with. This allows maximum flexibility with little impact on other team members.

If system resources do not allow multiple servers to be created, the next best scenario would have each team member with their own application. The ability to leverage this depends on the source control software being used. It is possible, in many cases, for team members to share a common source base, but generate .war and .jar files by different names and install the code as different applications in an IAS instance. It is possible and reasonable, in this scenario, for the team members to share DB connection ids (pools). Care would need to be taken to coordinate across the team any requirement to change a DB connection id (pool) configuration. This coordination restriction can be avoided and maximum flexibility achieved by each team member having their own DB connection ids (pools). This is true, even if the DB connection ids configurations would be duplicated across team members (only the connection id name itself would have to be unique).

Regarding other environments (quality, performance, production, etc.), they all tend to be a variation of the development environment, and the same recommendations would hold true.

### 4.1.5 Performance

#### Memory

#### Threads

#### Database

**Web Performance Advisor**

The Web Performance Advisor tool is a great way to view all the attributes that can have an effect on the performance of this application server. The Web Performance Advisor provides a way to review, evaluate, and modify the performance attributes for your Web environment. A Web environment is a grouping of related Web server, application servers, and operating system settings that form a Web solution. A Web environment is typically made up of a single WebSphere Application Server instance or an Integrated Web Application Server. Each server has its own set of attributes that needs to be evaluated and adjusted. The Web envi-
IBM Integrated Web Application Server for i

The IBM Integrated Web Application Server for i environment also includes any associated HTTP Web servers and system attributes that can have a direct affect on performance. Each component of the Web environment contains attributes that can affect the overall performance of this environment.

The Web Performance Advisor is integrated as part of the IBM Web Administration GUI interface. Enter the port 2001 interface. Select the server you wish to work with from the server drop down list under the Application Servers tab. In the left navigation area select the Web Performance Advisor link.

The Web Performance Advisor tool is a very powerful tool that may have a positive effect on the performance of this Web environment. Web Performance Advisor does not consider all possible settings and configurations and so does not replace the review of a performance professional. The WPA tool does however give a good base line setting. Please review the Web Performance Advisor Information Notice, and then click OK to continue using the tool.
Figure 79 - Web Performance Advisor Information Notice
Figure 80 - Web Performance Advisor

The Web Performance Advisor is made up of multiple components to help you tune the performance of your system and Web environment. On the introduction page, you are provided a quick, easy-to-read, high-level view of your system and Web environment performance evaluations. From here, you can determine if further tuning is required.

- The system performance attribute information section shows your hardware configurations and other system attributes related to performance along with an icon to indicate a green, yellow, or red performance rating. Click **Manage system attributes** to tune your performance with the Advisor function.

- The Web environment section defines the Web environment currently under evaluation, lists the components of this Web environment, and displays an icon to indicate a green, yellow, or red performance rating for each component. Click **Manage attributes** to tune your performance with the Advisor function. The Advisor function allows you to manage system attributes and to manage Web environment attributes.

System attribute information made up of the physical and logical resources that have been allocated to the system and partition, the system values that can have a direct effect on Web performance, TCP/IP settings, and PTF information including the PTF groups and the individual product PTFs for only the products that are used in a Web environment.
The System Resources tab displays the physical performance attributes of the system currently being used. Shows the number of active and virtual processors, the model and processor feature, along with the CPW for the entire system. Note: the CPW is not the value for the partition if you are running in a LPAR, but rather the CPW for the entire system. The attribute that is most useful is the Disk information. The Disk unit condition field shows if the disk array is in good working order. If the cache write battery is not working or a unit within a RAID array is not working correctly, the performance on the system can be adversely affected.

In the picture above, you can see a number of attributes with green check marks. The WPA code has determined that these attributes are set to values that are ‘good’. There is one attribute, with a yellow ‘!’ icon. The value for this attribute has been determined that it could be ‘better’. Click on the Advise link to launch the advise interface.

---

**Figure 81 - Web Performance Advisor - System Attributes, System Resources tab**

<table>
<thead>
<tr>
<th>System Information</th>
<th>System model: 595</th>
<th>Processor feature: 7903</th>
<th>System CPW: 92000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Information</td>
<td>Active processors: 2.0</td>
<td>Virtual processors: 2</td>
<td>Processor sharing: Shared capped</td>
</tr>
<tr>
<td>Disk Unit Information</td>
<td>Disk units: 6</td>
<td>Disk unit condition: Good</td>
<td></td>
</tr>
</tbody>
</table>

---
The Advise interface provides an explanation of what this performance attribute is, how it has an effect on the performance. The best part, it also provides a recommendation on what the 'best' value for this attribute is. This recommendation has been determined by the experts on the IBM i Performance team.
Figure 83 - Web Performance Advisor - Performance Settings tab

The Performance Settings tab displays a number of System values that have a direct impact on the performance of the Web environment. It also displays the TCP/IP buffer size.

Click on OK or Apply to save any modifications you made to any of these values. Changing one of the System Value attributes on this form is the same as using the WRKSYSVAL green screen interface. This interface just filters only the system values that have a direct impact on performance in the Web space.

Web Environment section contains attribute information for the Integrated Web Application Server instance. The Web performance attributes being tuned include the IAS JVM settings, system and server resource settings, database connection, and additional application server settings. The associated HTTP server is also displayed and is accessible.
Select the instance you wish to work with, click on the **Manage attributes** button to launch the manage Web environment interface.

The JVM Settings tab combines all the attributes related to the JVM container this application server is running in. View or modify these attributes. Click on the...
IBM Integrated Web Application Server for i

_Advise_ link to learn valuable information about what each attribute is and how it effect the performance on the application server.

**Figure 86 - Web Performance Advisor - Resources tab**

The Resources tab provides a view of the memory and threads that are available to the application. This combines the JVM heap information with the System memory pool information. This provides a complete view of what the memory and thread for the application are.

**Figure 87 - Web Performance Advisor - Database Connections tab**
The Database connection tab shows all the database connections that have been created for this application server. To work with the performance attributes for a specific database connection, click on the Database Connection ID value. This is a link to the Manage database connection attributes page.

Figure 88 -Web Performance Advisor - Manage Database Connection Attributes

Update the database pooling options. Click on the Advise link to get an explanation for each attribute along with the recommended value. Enable database pooling along with the correct connection pooling values.

Once you have competed adjust the performance attributes for the application server, you will need to restart the application server before these changes will go in to effect.

4.1.6 Monitoring

Health Checker

Collection Services

SST and JAVAGCTOOLS
4.1.7 Security

Required authorities

Iwencoder command

SSL

Environment setup: SSL Enablement for the IBM Integrated Web Application Server

By default the IBM Integrated Web Application Server for i listens on a non-
secure port. In order to change the server to make the port SSL enabled, the fol-
lowing steps need to be taken:

- Setup/Configure the *SYSTEM certificate store in Digital Certificate
  Manager (DCM)
- Create a Local Certificate Authority (CA).
- Create a locally signed system certificate within DCM
- Enable SSL within the HTTP Apache server associated with Integrated
  Web Application Server in the IBM Web Administration GUI interface.
- Within DCM, assign a system certificate to the application ID assigned to
  the associated HTTP server.

NOTE: This document will assume the use of a locally created system certificate.
If a certificate from a well known CA is needed, please refer to the following In-
formation Center Document:

http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/rzahu/rzahurza
hu437completenewstore.htm?tocNode=int_227379

Create/Configure the *SYSTEM certificate store in DCM

In order to access the Digital Certificate Manager (DCM) we first need to ensure
that the HTTP ADMIN server is up (STRTCPSVR SERVER(*HTTP)
HTTPSVR(*ADMIN) ). We then need to bring up a Web browser and access the
GUI interface using the following address (replace systemname with the name or
IP address of your IBM i): http://systemname:2001/

You should see a screen similar to the following:
Once on the i5/OS Tasks menu click on 'Digital Certificate Manager'. You should see a screen similar to the following:
To bypass the initial signon screen and go directly to Digital Certificate Manage the following URL can be used:

http://systemname:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0

In order to judge what needs to be created, we first should click on the ‘Select a Certificate Store’ button. If you see a screen that looks similar to the following, proceed to Section A - (Creating *SYSTEM Store, Local CA, and Local server certificate):

Figure 91 - Welcome page Digital Certificate Manager

Figure 92 - Select a Certificate Store – No defined stores
IBM Integrated Web Application Server for i

If your screen looks like the following, proceed to Section B - (Creating a Local Certificate Authority and a Local Server Certificate inside and existing *SYSTEM store).

![Select Certificate Store](image)

**Figure 93 - Select Certificate Store - user existing *SYSTEM store**

**Section A - (Creating *SYSTEM Store, Local CA, and Local server certificate)**

The options in ‘Select a Certificate Store’ show that a Local Certificate Authority (CA) store and a *SYSTEM store are missing and need to be created. Fortunately, the DCM wizard allows us to create a CA certificate, a *SYSTEM store, and a locally signed server certificate in the store all from one wizard.

1. Click on ‘Create a Certificate Authority (CA)’ on the left-hand menu bar. You will be presented with the ‘Create a Certificate Authority (CA)’ form to fill out as follows (the required fields are described below):

   - **Key Size**: Keep this the default of 1024
   - **Certificate Store Password**: set a password for the CA store
   - **Confirm Password**: confirm the password

   - **Certificate Authority (CA) name**: This is the name of the CA that will sign certificates (i.e. IBM Test CA)
   - **Organization Name**: Your organization name (i.e. IBM)
   - **State or Province**: State/Province name, must contain 3 characters minimum (i.e. MIN)
   - **Country or region**: Country/Region name (i.e. US)

   - **Validity period of Certificate Authority (CA) (2-7300)**: Usually this is set to 7300 to make the certificate last as long as possible without the need for renewal

   Click ‘Continue’
IBM Integrated Web Application Server for i

2. The next screen will present you with the option to ‘Install Certificate’. This allows you to install the CA certificate into the browser, so that it will trust the CA. Feel free to do this if you’d like, it is not required. Then click ‘Continue’

3. On the next screen you should see the ‘Certificate Authority (CA Policy Data)’ form with the following parameters:

   **Allow creation of user certificates:** *Keep the default of ‘Yes’*

   **Validity period of the certificates that are issued by this Certificate Authority (CA):** *In order to make the certificates last as long as possible set this to 2000 days*

   Click ‘Continue’

4. You should get a screen that looks like the following:

   ![Policy Data Accepted](image)

   **Figure 94 - Policy Data Accepted**

   Click ‘Continue’

5. We now should be at the ‘Create a Server or Client Certificate’ wizard screen. Fill out the form with the following information (we’ll only need the required fields):

   **Key size:** Keep this the default of 1024

   **Certificate Label:** This is the label the certificate will have within DCM (i.e. Test cert)

   **Certificate Store password:** Set a password for the *SYSTEM store

   **Confirm password:** Confirm your password

   **Common Name:** This is the URL that people will type to get to your site (i.e. Systemmane.mycompany.com) **NOTE:** No http:// is required
IBM Integrated Web Application Server for i

**Organization Name:** Your company/organization name (i.e. IBM)

**State or Province:** State/Province name, needs 3 characters minimum (i.e. MIN)

**Country or region:** Country/Region name (i.e. US)

**NOTE:** Skip the ‘Subject Alternative Name’ section at the bottom.

Click ‘Continue’

6. You should receive a confirmation message that looks like the following:

![Figure 95 - Confirmation message](image)

**Figure 95 - Confirmation message**

7. Click ‘Continue’ at the bottom of the page, because we don’t have an application to assign the certificate to yet. Do the same for the next screen, as well as, for the CA.

You should now see a screen similar to the following:

![Application Status message](image)

**Figure 96 - Application Status message**

8. Click ‘OK’, you’ve now completed the DCM setup

9. On the next screen, you’ll be asked if you would like to create an *OBJECTSIGNING certificate store, click ‘Cancel’ (we do not need to create this). You’ve now successfully created a local server certificate.
IBM Integrated Web Application Server for i

10. Skip over sections B and C below.

Section B - (Creating a Local Certificate Authority and a Local Server Certificate inside and existing *SYSTEM store).

The *SYSTEM store appears to be created and is showing up in the ‘Select a Certificate Store’ screen. We will now create a Local Certificate Authority (CA) certificate, which will allow us to create Local Server Certificates.

1. Click on the ‘Create a Certificate Authority (CA)’ link on the left menu bar. You should then see a ‘Create a Certificate Authority (CA)’ form. Here is a description of each of the required fields:

   - **Certificate Store password:** This is the password for the CA certificate store, it is recommended to keep this the same as the *SYSTEM store for simplicity sake.
   - **Confirm password:** Confirm the password set.
   - **Certificate Authority (CA) name:** This is the CA name that will sign the local certificates (i.e. IBM CA).
   - **Organization Name:** The name of your organization (i.e. IBM).
   - **State or province:** The name of your state or province, requires a minimum of 3 characters (i.e. MN).
   - **Country or region:** The name of your country or region (i.e. US).
   - **Validity Period of Certificate Authority (CA):** It is recommended to set this to the max of 7300 days to ensure the CA certificate lasts as long as possible.

   Click ‘Continue’

2. You may choose to click on ‘Install certificate’, which will install the CA certificate into the browser so that it is trusted. This is an optional step, click ‘Continue’ when finished.

3. On the next screen you should see the ‘Certificate Authority (CA Policy Data)’ form with the following parameters:

   - **Allow creation of user certificates:** Keep the default of ‘Yes’
   - **Validity period of the certificates that are issued by this Certificate Authority (CA):** In order to make the certificates last as long as possible set this to 2000 days

   Click ‘Continue’
IBM Integrated Web Application Server for i

4. You should get a screen that looks like the following:

![Select Applications to Trust this Certificate Authority (CA)](image)

**Figure 97 - Select Applications to Trust**

Click 'Select All' then click 'Continue' down at the bottom.

5. On the next screen, you'll be asked if you would like to create an *OBJECTSIGNING certificate store, click 'Cancel' (we do not need to create this). You've now successfully created a Certificate Authority (CA).

6. Now that the CA certificate has been created we'll need to create a new local server certificate, signed by the CA. To do this we first need to click on the ‘Select a Certificate Store’ and select the radio button next to *SYSTEM. Click 'Continue’. Sign into the *SYSTEM store (or if needed the ‘Reset Password’ button can be used if the password is forgotten).

8. On the left-hand menu bar click ‘Create Certificate’. Select 'Server or client certificate and click 'Continue'. We'll then want to select 'Local Certificate Authority (CA)' and click 'Continue'.

9. You should be presented with the Create Certificate Form. Fill out the form as follows:

- **Key size**: Keep this the default of 1024
- **Certificate Label**: This is the label the certificate will have within DCM (i.e. Test cert)
- **Common Name**: This is the URL that people will type to get to your site (i.e. Systemmane.mycompany.com) **NOTE**: No http:// is required
- **Organization Name**: Your company/organization name (i.e. IBM)
- **State or Province**: State/Province name, needs 3 characters minimum (i.e. MIN)
- **Country or region**: Country/Region name (i.e. US)

**NOTE**: Skip the ‘Subject Alternative Name’ section at the bottom

Click 'Continue'

10. You should see a screen similar to the following:
11. At this time we don’t have an application to assign the certificate to, down at the bottom of the screen click ‘Continue’. Click on the ‘OK’ button on the next screen. You have now created a local certificate in the *SYSTEM store that we can use to secure the HTTP server associated with the Integrated Web Application Server for i.

12. Skip over section C below.

**Section C – (Creating a Local Server Certificate inside an existing *SYSTEM store with an existing Local Certificate Authority (CA))**

Based on what we see on the ‘Select a Certificate Store’ menu we can see that we already have a *SYSTEM store created, as well as, a Local Certificate Authority (CA). Since both of these steps have been done all we need to do is create a server certificate, which we can use to secure the HTTP server associated with the Integrated Web Application Server for i.

1. First we’ll need to click on the ‘Select a Certificate Store’ and select the radio button next to *SYSTEM. Click ‘Continue’. Sign into the *SYSTEM store (or if needed the ‘Reset Password’ button can be used if the password is forgotten).

2. On the left-hand menu bar click ‘Create Certificate’. Select ‘Server or client certificate’ and click ‘Continue’. We’ll then want to select ‘Local Certificate Authority (CA)’ and click ‘Continue’.

3. You should be presented with the Create Certificate Form. Fill out the form as follows:

   **Key size:** Keep this the default of 1024
   
   **Certificate Label:** This is the label the certificate will have within DCM (i.e. Test cert)
   
   **Common Name:** This is the URL that people will type to get to your site (i.e. Systemmane.mycompany.com)  
   **NOTE:** No http:// is required
   
   **Organization Name:** Your company/organization name (i.e. IBM)
   
   **State or Province:** State/Province name, needs 3 characters minimum (i.e. MIN)
   
   **Country or region:** Country/Region name (i.e. US)
IBM Integrated Web Application Server for i

**NOTE:** Skip the ‘Subject Alternative Name’ section at the bottom
Click ‘Continue’

4. You should see a screen similar to the following:

   ![Certificate placed in store message](image)

   **Figure 99 - Certificate placed in store message**

5. At this time we don’t have an application to assign the certificate to. Click ‘Continue’, down at the bottom of the screen. Click on the ‘OK’ button on the next screen. You have now created a local certificate in the *SYSTEM store that you can use to secure the HTTP Apache server associated with the Integrated Web Application Server.

*Enable SSL within the associated HTTP Apache server*

1. Now that we have our *SYSTEM store configured and have a server certificate created we need to enable the server for SSL.

2. First we’ll need to return to the i5/OS Tasks menu page:  
http://systemname:2001

3. Click on ‘IBM Web Administration for i5/OS’

4. Click on the ‘Manage’ tab at the top, then click on ‘HTTP Servers’. From the server drop-down box select the HTTP Server associated with the Integrated Web Application server (in this example INTAPPSVR) :
5. On the left-hand menu click on ‘Security’

6. On the Security menu screen select ‘Enabled’ for SSL. Your screen should now look like the following:

![Figure 101 – SSL with Certificate Authentication](image)

7. Notice now that you have a warning that says ‘Server certificate application name’ in red. Click on the drop down that has ‘or...’. Click on the default application name given, as follows:
8. Once you’ve selected the application name from the drop down, it will place the name in the box to the left of the ‘or...’ Drop down. Scroll down a little and you should see the ‘HTTPS_PORT environment variable’. Put in the port that the INTAPPSVR server is listening on (this can be found in the HTTP configuration for the server under ‘Display Configuration on the left-hand menu bar. See the example below:).

9. Click ‘OK’
IBM Integrated Web Application Server for i

We've now created the QIBM_HTTP_SERVER_INTAPPSVR application ID inside DCM. We'll be able to assign a certificate to this application ID which will complete our SSL enablement. In the INTAPPSVR Apache server configuration we now have the following three Apache directives ('####' is the port number that INTAPPSVR is configured to listen on):

- SSLEngine On
- SSLAppName QIBM_HTTP_SERVER_INTAPPSVR
- SetEnv HTTPS_PORT ####

**Assigning a Certificate to the INTAPPSVR Application ID in DCM**

We now have the following things setup:

- The *SYSTEM store is configured
- A Local Certificate Authority (CA) is created
- A local server certificate signed by the Local CA
- The INTAPPSVR Apache server is configured for SSL

The last remaining step is to assign the local server certificate to the application ID that was created when we enabled SSL for the INTAPPSVR HTTP Apache server.

1. We'll need to return to the i5/OS Tasks menu (http://systemname:2001). Click on 'Digital Certificate Manager'.

2. Click on the ‘Select a Certificate Store’ button and sign into the *SYSTEM store.

3. On the left menu click on ‘Fast Path’, then click on 'Work with server and client certificates' under ‘Fast Path’.

4. You should now see the local server certificate listed that was created previously. Click on the radio button next to the certificate name, and then click on the 'Assign to Applications' button.

5. You should see near the bottom of the page the following check box:

   ![](figure104.png)

   **Figure 104 - Assign to Application prompt**

6. Check the box next to the QIBM_HTTP_SERVER_INTAPPSVR application ID and then click on 'Continue'.
IBM Integrated Web Application Server for i

You should receive the following confirmation:

![Message: The applications you selected will use this certificate.](image)

**Figure 105 - Assign Application confirmation message**

7. We now have completed the setup and the INTAPPsvr server is enabled for SSL. To complete the process we'll need to Stop then Start the INTAPPsvr HTTP server for the changes to take place. From the Manage HTTP Servers tab, click on the Stop button for the INTAPPsvr server. Once the server is in a ‘Stopped’ status click the start button.

Once the INTAPPsvr server is restarted we now can access the IBM Integrated Web Application Server for i applications via the SSL enabled port.

**Notes**

1. Programmatic enablement of SSL only enables the internal server port. Apache server is not enabled.
2. If problem encountered during enablement or disablement of SSL, server is left in an unknown (probably unusable) state. Best course of action is to remove the server and recreate.
5. Appendix

A. Appendix A – References


The STG ISV Enablement organization is a resource that partners can utilize to not only help with direction, strategy and technical know how for using the IBM i Integrated Application Server, but as a partner for you to use on any IBM i, STG or SWG technology that you need some assistance on to make your application more feature rich and competitive in the market place. We offer technical collateral to get you started, virtual access to systems for testing your enhanced applications, as well the technical resources that are part of your team as you move to integrate and use these new technologies. For starters, visit the "Leverage IBM Systems ISV resources to win" web page at https://www-304.ibm.com/jct09002c/partnerworld/wps/pub/systems/whyibm/programs.
B. Appendix B  Glossary

**Bundle** A bundle is the level at which components are declared to the OSGi Service Framework. A bundle is a JAR file with a bundle manifest file named MANIFEST.MF. The bundle manifest describes the bundle to the service framework and enables a bundle to consume and/or provide packages and services from/to other bundles.

**Component** A component is an individual Java object or an archive of Java objects. A component in OSGi implements a java interface, i.e. a Service or Service Interface. Applications in OSGi are built from collaborating components coupled only via their service interface.

**Eclipse** A universal platform for integrating components to create applications. The eclipse framework is built on the OSGi Service Framework. (www.eclipse.org)

**Exploiter** This term refers to a consumer of IAS, i.e. the product or platform operating system that embeds and exploits IAS.

**Feature** A products components are grouped together into features. A feature is the smallest unit of separately downloadable and installable functionality.

**Fragment** Part of a plug-in that is optional, allowing it to be installed, uninstalled, or updated independently from the rest of the plug-in. Fragments can only add function/content to plug-ins. They cannot override that which the plug-in already contains. A fragment is associated with a specific primary plug-in.

**IAS** (IBM Integrated Web Application Server) A Web based transaction server capable of high performance and very low memory. Primarily target is for the low use or simple application not requiring high scaling and HA.

**IWS** (IBM Integrated Web Services Server) Web Services runtime engine (based on AXIS2) integrated as part of the IAS framework. Wizard within the Web Admin GUI provide a point a click creation of RPG & COBOL programs directly into a Web Service. (http://www-03.ibm.com/systems/i/software/iws/)

**CL** (IBM Common Logging) A set of API wrappers which allow developers to code to a common set of logging/tracing APIs.
IBM Integrated Web Application Server for i

IES (IBM Eclipse SDK) The authorized and approved line up of Eclipse components used by IBM teams to distribute Eclipse based products. (http://ies.ottawa.ibm.com/ies)

Indus The release of the Integrated Solutions Console that runs on IAS.

JACC (Java Authorization Contract for Containers) a Java specification (JSR-115) that defines new java.security.Permissions classes to satisfy the Java 2 Platform Enterprise Edition (J2EE) authorization model. The JACC specification defines the binding of container access decisions to operations on instances of these permission classes.

JDBC (Java Database Connectivity) API is a standard SQL database access interface.

JNDI (Java Naming and Directory Interface) enables Java applications to access different naming and directory services using a common API.

JSF (Java Server Faces) is a server side user interface framework for Java web applications. It is designed to ease the burden of writing and maintaining applications that run on a Java Application Server and render their UI's back to a target client.

JSR 127 (Java Specification Request 127) is a Java Community Process specification for Java Server Faces (JSF)

JSR 168 (Java Specification Request 168) is a Java Community Process (JCP) portlet specification that defines a set of Java APIs to enable interoperability between portals and portlets.

JSR 172 (Java Specification Request 172) is a Java Community Process specification to define an optional package that provides standard access from J2ME to web services

JWL (Java Widget Library) An extended widget set usable by portal and portlet programming. It provides Java Script client-side processing to JSF. Updating the view locally on the client saves round trips to the server, shortens the response time, and improves the user experience.

MUSE (Apache MUSE) An implementation of MUWS.

OSGI (Open Services Gateway initiative) an independent, non-profit corporation working to define open specifications for the delivery of services to networked small-memory devices. The OSGi Service Platform specifications describe the software deployment and configuration management architecture.

Plug-in A plug-in is the level at which components are declared to the Eclipse framework. A plug-in is a JAR file with a plug-in manifest file named plugin.xml. The plug-in manifest describes the plug-in to the framework and enables a plug-in to consume and/or provide extensions from/to other plug-ins.

RCP The Eclipse Rich Client Platform can be described as the minimal set of plug-ins needed to build a platform application with a UI. For more information, go to www.eclipse.org/rcp/.
Service A service is a self-contained component accessible via a service interface. Services are registered with the OSGi framework, making them available to others. This is sometimes called publishing a service. Components can acquire services by querying the framework. Components must release the services back to the framework when finished.

Servlet A compiled java class that exposes web-page-like functionality. A servlet is conceptually similar to a CGI executable, except that it is managed by a container and can maintain state between client requests.

SMF (Service Management Framework) an Implementation of the OSGi standard which includes OSGi specification implementation plus IBM value-added services.

WAB (Web Application Bundle, a.k.a. OSGi Bundle) is a Web application packaged as a bundle that can be deployed in an OSGi framework.

WAR (Web Archive) is a packaged Web application that can be exported to test, publish, and deploy the resources developed within a Web project.

WebSphere Application Server is a transaction server that features high performance and an extremely scalable transaction engine capable of handling high-volume, secure transactions and Web services. Application servers are middleware that handle application functions between users and the back-end applications and databases of a business. The WebSphere Application Server is Java-based. The basic elements of a Java-based application server are servlets, JSPs and JavaBeans.

Web Container A runtime environment for web components that typically includes security, concurrency, life-cycle management, transaction, deployment, and other services. A web container is provided by a web server or J2EE server.

WSDL (Web Service Description Language) An XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.

WSDM/MUWS (Web Services Distributed Management: Management Using Web Services) A specification that defines how to represent and access the manageability interfaces of IT resources as Web services.

WSRF (Web Services Resource Framework) a generic and open framework for modeling and accessing stateful resources using web services.
C. Appendix C Commands

Notes:
- commands for each server can be found in the System i IFS directory, in one of two places:
  - /QIBM/ProdData/OS/OSGi/LWI71/bin – It is suggested that scripts be executed from the other location listed here. Executing scripts here requires that additional arguments be specified when calling the script (i.e. the IAS instance would have to be specified when executing lwistart.sh).
  - /www/<IAS server name>/lwi/bin/ - it is suggested that the commands be executed from this location. It prevents the caller from having to specify additional arguments as part of calling the script. For instance, the name of the IAS instance does not have to be specified on lwistart.sh. The descriptions that follow in this appendix assume that the scripts will be executed from this location.
- (Scheduled to change; update required) Prior to executing a command in QShell, one of the following needs to be executed (only once per QShell session) to set the Java home:
  - export JAVA_HOME=/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit, or
  - lwiSetJavaHome.sh /QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit
- TBD: which of the following scripts should be included?

launchlwi.sh

Purpose
Starts an IAS instance. Note: this script is not intended to be executed directly. It is executed as part of the lwistart.sh command, which is the recommended script for starting IAS instances.

Usage
lwistart.sh [-help]

Usage Notes
- help Displays command usage

Command Logic
1. executes setupCmdLine.sh
2. changes the current directory to /www/<IAS_name>/lwi/runtime/core
3. initializes various variables:
   - UTIL_JARS - points to the jar files in directory /www/<IAS name>/lwi/lib
   - JAVAOPTS – uses the options in the /www/<IAS_name>/lwi/conf/overrides/i5javaopts.javaopt file
   - LWI_DEBUG_ARGS – by default, this is commented out. If uncommented, would initialize to “-Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,server=y,address=8000,suspend=n”
IBM Integrated Web Application Server for i

4. verifies that the /www/<IAS_name>/lwi/conf/admin/adminAuth.properties file has read permission. If not, it is granted.
5. initializes variables LD_LIBRARY_PATH and LIBPATH to /www/<IAS_name>/lwi/native and exports them
6. builds the command line to launch the IAS instance; this is the command line to execute the java class com.ibm.lwi.LaunchLWI
   - options:
     - some of the variables previously initialized in this command (JAVAOPTS, LWI_DEBUG_ARGS, UTIL_JARS)
     - -Xbootclasspath
       - RCPBOOTCP_JARS, initialized in setJars.sh, which is called by set-upCmdLine.sh
       - UTIL_JARS, previously initialized in this command
   - classpath:
     - jars from the /www/<IAS_name>/lwi/runtime/core path
       - eclipse/launch.jar
       - eclipse/startup.jar
       - rcp/eclipse/plugins/com.ibm.rcp.base_6.1.2.200801281200/launcher.jar
     - parameters: those passed into this lwiadminuser.sh script
7. execute the command line just built. The com.ibm.lwi.LaunchLWI class, among other tasks, loads the primary config file /www/<IAS_name>/lwi/conf/config.properties and the override config files in /www/<IAS_name>/lwi/conf/overrides
8. (verify that none of the OS400-unique logic is executed)

lwi64bit.sh
IBM internal use only.

lwiadminuser.sh
Purpose
Allows management of the admin user shipped with IAS, including locking, unlocking, resetting the number of failed attempts, and listing.

Usage
lwiadminuser.sh {-lock | -unlock | -reset} <user_name>
lwiadminuser.sh {–listLocked | -help }

Usage Notes
- -lock Lock the user that is specified by user_name.
- -unlock Unlock the user that is specified by user_name.
- -reset - Reset the number of failed attempts for the user specified by user_name.
- -listLocked Lists the locked users.
IBM Integrated Web Application Server for i

- **help** Displays command usage

**Command Logic**
1. executes setupCmdLine.sh
2. initializes RUNTIME_DIR variable to /www/<IAS_name>/lwi/runtime/core
3. executes the class com.ibm.lwi.utils.cli.AdminUser (what does this do?)
   - **classpath:**
     - some of the variables in the setJars.sh, which is called by setupCmdLine.sh (LWIUTILS_JAR, LWIDEPLOY_JAR, WEBCONTAINER_JAR, SHARDBUNDLE_JAR)
     - an additional jar of /www/<IAS_name>/lwi/lib/lwinlall.jar (for national language support)
   - **parameters:** those passed into this lwiadminuser.sh script

**lwiClasspath.sh**

**Purpose**
To add or remove boot classpath entry to/from the IAS instance.

**Usage**
lwiClasspath.sh {-add|-remove} classpathEntry
lwiClasspath.sh {-list | -help}

**Usage Notes**
  - **-add** Add a classpath entry to the IAS instance
  - **-remove** Remove a classpath entry from the IAS instance
  - **-list** Lists the classpath entries configured for this IAS instance
  - **-help** Displays command usage.

**Command Logic**
1. executes setupCmdLine.sh
2. executes the class com.ibm.lwi.config.LwiClasspath (what does this do?)
   - **classpath:**
     - /www/<IAS_name>/lwi/conf/configLWI.jar
     - LWI_NLS_LIB, previously set up in setupCmdLine.sh
   - **Java arguments:** -Dlwi.dir=/www/<IAS_name>/lwi
   - **parameters:** those passed into this lwiClasspath.sh script

**lwiCopyConfigFile.sh**

**Purpose**
Copies a configuration file into the appropriate directory of the IAS Instance directory structure. This command is useful for exploiters who do not want to copy the file into the directory themselves. The IAS instance needs to be restarted for the changes to take effect.

**Usage**
lwiCopyConfigFile.sh {-properties | -jar | -module | -jaasconfig} <configFile> [-help]
IBM Integrated Web Application Server for i

Usage Notes

- **properties** Indicates that the `configFile` specified contains additional or
override java properties for the IAS instance. The specified file will be
copied to the conf/overrides directory of the IAS instance.

- **jar** Indicates that the `configFile` specified is a utility jar which should be
added to the boot classpath of the IAS instance. The specified file will be
copied to the lib directory of the IAS instance.

- **module** Indicates that the `configFile` specified is a native module. The
specified file will be copied to the native directory of the IAS instance.
Applications will be able to load the module by calling
`java.lang.System.loadLibrary`.

- **jaasconfig** Indicates that the `configFile` specified is a JAAS login con-
figuration file. The specified file will be copied to the secu-
ritY/jaas/addtlconfig directory.

- **help** Displays command usage

- **configFile** Absolute path to the config file.

Command Logic

1. executes `setupCmdLine.sh`
2. executes the class `com.ibm.lwi.config.CopyConfigFile` (what does this do?)
   a. classpath:
      i. `/www/<IAS_name>/lwi/conf/configLWI.jar`
      ii. LWI_NLS_LIB, previously set up in `setupCmdLine.sh`
   b. Java arguments: `-Dlwi.dir=/www/<IAS_name>/lwi`
   c. parameters: those passed into this `lwiCopyConfigFile.sh` script
3. echoes SUCCESS if the command executed successfully

`lwiCreateExtension.sh`

Purpose

Add a directory which contains plug-ins and features to the IAS startup environ-
ment. The plug-ins in the directory will automatically be loaded the next time the
IAS instance is started. The IAS instance does not have to be active (running) for this command to work. If the instance IS active, it will need to be restarted for the plug-ins to be loaded.

To deploy plug-ins and features to an active IAS instance, see Section
`lwiplugin.sh`.

Usage

`lwiCreateExtension.sh <pathToExtension> [-help]`
IBM Integrated Web Application Server for i

Usage Notes

- **-help** Displays command usage
- **pathToExtension** must be a directory containing the plug-ins/features to be deployed.
- The directory specified by **pathToExtension** must contain a subdirectory named **eclipse** which itself contains one or both of the subdirectories **plugins** and/or **features**.
- The plug-ins/features to be deployed must conform to the Eclipse link format.

Usage Examples

If the plug-ins are located in `/opt/IBM/xyz/eclipse/plugins`, then:

```
lwiCreateExtension.sh /opt/IBM/xyz.
```

Command Logic

1. executes **setupCmdLine.sh**
2. executes the class **com.ibm.lwi.config.CreateExtension** (what does this do?)
   a. classpath:
      i. `/www/<IAS_name>/lwi/conf/configLWI.jar`
      ii. LWI_NLS_LIB, previously set up in **setupCmdLine.sh**
   b. Java arguments: `-Dlwi.dir=/www/<IAS_name>/lwi`
   c. parameters: those passed into this **lwiCreateExtension.sh** script
3. echoes SUCCESS if the command executed successfully, otherwise exits with the java execution return code

**lwiencoder.sh**

Purpose

The **lwiencoder** command is used to encrypt passwords that are stored in the IAS configuration files. Note: the utility can not encode passwords contained in XML files or other files that contain open and close tags. To ensure sensitive information, such as passwords, are secure, a deployer must restrict access to the configuration file so that only the appropriate users can read the file. This would typically be done by setting the file permissions of the configuration file or file system such that only authorized users could access it.

Usage

```
```

Usage Notes

- **-filename** Use of the **-filename** flag without the **-keylist** argument will cause all properties in the file to be encoded. The filename path can be relative to the current working directory, or a fully qualified path.
IBM Integrated Web Application Server for i

- **keylist** To encrypt specific properties, use the –keylist flag. Multiple properties can be specified using commas for separation.

- **backup** To backup the original file, use the –backup flag. The backup will be created in the same location as the original.

- **crypt** value must be one of the supported algorithms: aes (default), xor or custom. If custom is used, and –cryptclass option is not, IAS will try to load com.ibm.lwi.security.crypto.CustomPasswordEncryptionImpl.class.

- **cryptkey** specifies a key word to be used for encryption (i.e. mysecret, or "My Secret Key").

- **cryptclass** is only used if –crypt is custom. Whatever class is specified here must implement the com.ibm.lwi.security.crypto.CryptPasswordEncryption interface and must be located in the java classpath (or be part of any jar placed in the /www/<IAS_name>/lwi/lib directory).

- **help** Displays command usage

If no flags are specified, a usage message is displayed

Flags can be specified in any order

**Usage Examples**

- lwiencode.sh -filename myProperties.txt
- lwiencode.sh -filename "My Properties.txt"
- lwiencode.sh -keylist key1,key2 -filename myProperties.txt
- lwiencode.sh -filename myProperties.txt –backup
- lwiencode.sh -filename myProperties.txt –crypt aes
- lwiencode.sh -filename myProperties.txt –crypt custom
- lwiencode.sh -filename myProperties.txt –cryptkey mysecret
- lwiencode.sh -filename myProperties.txt –cryptkey "My Secret Key"
- lwiencode.sh -filename myProperties.txt –crypt custom –cryptclass my-company.MyCustomPasswordEncryption.class

**Exit Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The command executed successfully</td>
</tr>
<tr>
<td>1</td>
<td>Couldn’t proceed because of missing required arguments</td>
</tr>
<tr>
<td>2</td>
<td>Illegal usage</td>
</tr>
<tr>
<td>3</td>
<td>Can’t accept null values</td>
</tr>
<tr>
<td>4</td>
<td>The comma separated list of property keys is empty</td>
</tr>
<tr>
<td>5</td>
<td>The command encountered an error while loading properties file</td>
</tr>
<tr>
<td>6</td>
<td>Command encountered an error while storing encoded values</td>
</tr>
<tr>
<td>7</td>
<td>One or more of the specified property keys not found in the file</td>
</tr>
<tr>
<td>8</td>
<td>One or more of the property values are already encoded</td>
</tr>
<tr>
<td>9</td>
<td>Properties file is empty</td>
</tr>
<tr>
<td>10</td>
<td>Invalid properties key list</td>
</tr>
<tr>
<td>11</td>
<td>Encoding error.</td>
</tr>
</tbody>
</table>

**Command Logic**

1. executes setupCmdLine.sh
IBM Integrated Web Application Server for i

2. Initializes UTIL_JARS variable to point to the jars in
   /www/<IAS_name>/lwi/lib
3. Initializes JAVA_PROGRAM to
   /QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit/bin/java
4. executes the class com.ibm.lwi.utils.cli.PropFilePasswordEncoder (what
does this do?)
   a. classpath:
      i. UTIL_JARS, previously initialized in this script
      ii. Variables previously initialized in the setJars.sh script, previously
called by setupCmdLine.sh (SHAREDBUNDLE_JAR, LWIUTILS_JAR)
   b. parameters: those passed into this lwiencoder.sh script

IwiEncryptPwForWCT.sh

Purpose
Creates an encrypted password string to be used in the webcontainer.properties
file.

Usage
IwiEncryptPwForWCT.sh -pw <myPassword> [-help]

Usage Notes
   o  **myPassword** Specifies the “clear” password that is to be encoded
   o  **-help** Displays command usage

Command Logic
1. executes setupCmdLine.sh
2. changes the current directory to /www/<IAS_name>/lwi/runtime/core
3. executes the class com.ibm.lwi.utils.cli.EncryptPwForWebContainer (what
does this do?)
   a. classpath:
      i. LWI_NLS_LIB, previously initialized in setupCmdLine.sh
      ii. variables previously initialized in the setJars.sh script, previously
called by setupCmdLine.sh (SHAREDBUNDLE_JAR, LWIUTILS_JAR)
   b. parameters: those passed into this lwiEncryptPwForWCT.sh script

Iwlilog.sh

Purpose
Refreshes the logging configuration properties. This function is especially needed
when:
   o  Logger message levels need to be changed in order to obtain additional
diagnostic information.
   o  Additional logger levels need to be added to obtain additional diagnostic
information.
IBM Integrated Web Application Server for i

This command should be used when the IAS instance is active. A typical use case of how to refresh the logging configuration properties would be:

- Use the lwilog command to add an additional logger level to the logging configuration in order to obtain additional diagnostic information for a particular piece of code (this could also be done by editing the /conf/overrides/logging.properties file). Using the lwilog command to update the logging configuration properties results in the logging configuration being refreshed by default. If the logging configuration file is updated manually, run the lwilog --refresh command to refresh the logging configuration properties.

- Execute the application/code to generate the additional diagnostic messages.

Usage

- lwilog.sh [-help | -refresh]
- lwilog.sh --addLogger --name loggerName --level levelValue [-propsfile filename] [-norefresh] [-temp]
- lwilog.sh --removeLogger --name loggerName [-propsfile filename] [-norefresh]
- lwilog.sh --addHandler --name handlerName [propertyName propertyValue [-propsfile filename] [-norefresh] [-temp]
- lwilog.sh --removeHandler --name Handler [-propsfile filename] [-norefresh] [temp]

Usage Notes

- -help Displays the command usage
- -refresh Refreshes the logging configuration of the active IAS instance with the values in the logging configuration properties file (www/<IAS_name>/wli/conf/overrides/logging.properties)
- -addLogger Adds a new logger to the logging configuration (updates the logging configuration properties file) and by default refreshes the logging configuration. If the logger already exists then the logger level is updated.
  - -name Specifies the Logger name to add or modify (for example, com.ibm.lwi.test).
  - -level Specifies the Logger level (for example, FINEST).
  - -propsfile Optionally specify an alternative properties file (for example, myAlternative.properties).
  - -norefresh Optionally, does not refresh the logging configuration (loggers or handlers), although it WILL be refreshed the next time the IAS instance is started
  - -temp Optionally add a temporary logger without updating the logging configuration properties file. This logger will be in affect until the IAS instance is stopped.
- -removeLogger Removes a logger from the logging configuration (updates the logging configuration properties file) and by default refreshes the logging configuration
  - -name Specifies the Logger name to be removed (for example, com.ibm.lwi.test).
IBM Integrated Web Application Server for i

- `-propsfile` Optionally specify an alternative properties file (for example, myAlternative.properties).

- `-norefresh` Optionally does not refresh the logging configuration (loggers or handlers).

  - `-addHandler` Adds a new handler to the logging configuration (updates the logging configuration properties file) and by default refreshes the logging configuration.

    - `-name` Handler name to be added (for example, com.ibm.MyHandler).

    - `-propertyName` Handler properties. The specific handler properties that can be specified are (if these properties are not specified, then defaults must be provided by the handler class):
      - `-level` Level of logs to be registered by the handler (for example, INFO).
      - `-size` Determines the maximum size in bytes the log file can reach. If the log size is exceeded then the log files are cycled (for example, 4000000).
      - `-count` Defines the maximum number of log files that the logging system cycles between (for example, 6).
      - `-pattern` Defines the name of the log file to be created by this handler (for example, MyErrorLogFile-%g).
      - `-formatter` Defines the formatter used for the log file (for example, com.my.formatter).
      - `-append` Defines whether on platform restart, logs should be recorded in a new log file, or appended to the last log file used (for example, false).

    - `-propsFile` Optionally specify an alternative properties file (for example, myAlternative.properties).

    - `-norefresh` Optionally does not refresh the logging configuration (loggers or handlers).

- `-temp` Optionally add a temporary handler without updating the logging configuration properties file.

  - `-removeHandler` Removes a handler from the logging configuration (updates the logging configuration properties file) and by default refreshes the logging configuration.

    - `-name` Handler name to be removed (for example, com.ibm.MyHandler).

    - `-propsFile` Optionally specify an alternative properties file (for example, myAlternative.properties).

    - `-norefresh` Optionally does not refresh the logging configuration.

    - `-temp` Optionally remove a temporary logger without updating the logging configuration properties file.
**IBM Integrated Web Application Server for i**

**Command Logic**
1. executes setupCmdLine.sh
2. changes the current directory to /www/<IAS_name>/lwi/runtime/core
3. executes the class com.ibm.lwi.utils.cli.LoggingUtils
   a. classpath:
      i. LWI_NLS_LIB, previously initialized in setupCmdLine.sh
      ii. variables previously initialized in the setJars.sh script, previously called by setupCmdLine.sh (WEBCONTAINER_JAR, SHARDBUNDLE_JAR)
   b. parameters: those passed into this lwilog.sh script
4. echoes SUCCESS if the java executed successfully, otherwise exits with the java execution return code

**lwimapRole.sh**

**Purpose**
Performs the file-based management of the principal-to-role mapping that is shipped with IAS runtime.

**Usage**
- lwimapRole.sh –add –role roleName [-user user1,user2,…] [-group group1,group2,…] [-application applicationName]
- lwimapRole.sh –remove –role roleName [-user user1,user2,…] [-group group1,group2,…] [-application applicationName]
- lwimapRole.sh –purge –role roleName
- lwimapRole.sh -query criteria
- lwimapRole.sh -help

**Usage Notes**
- **-help** Displays command usage
- **-add** Specifies to add or create a role to principal or group mapping.
  - **-role** Specifies a role name which exactly corresponds to an application supplied role, such as the role-name value of a security-role element from the web.xml.
  - **-user** Specifies a comma-separated list of users which should be mapped to this role.
  - **-group** Specifies a comma-separated list of groups which should be mapped to this role.
  - **-application** An application identifier which is associated with this role. The application-to-role mapping is not used by the IAS runtime. Rather it allows an application running within the IAS runtime to retrieve a list of roles associated with the applicationName.
IBM Integrated Web Application Server for i

- **-remove** Specifies to remove a principal or group from the role mapping.
  - **-role** Specifies a role name which exactly corresponds to an application-supplied role, such as the role-name value of a security-role element from the web.xml.
  - **-user** Specifies a comma-separated list of users which should be removed from this role.
  - **-group** Specifies a comma separated list of groups which should be removed from this role.
  - **-application** Specifies the application identifier which should be removed from this role.

- **-purge** Completely remove a role and all of its associated mappings.
  - **-role** Specifies a role name which exactly corresponds to an application supplied role such as the role-name value of a security-role element from the web.xml.

- **-query** Query the current role mappings based on a given criteria.
  - **criteria** Criteria is one of:
    - getRoles
    - getRolesByUser username
    - getRolesByGroup groupName
    - getUsersByRole roleName
    - getGroupsByRole roleName
    - getRolesByApplication appName

**Command Logic**

Executes setupCmdLine.sh

Executes the class com.ibm.lwi.security.rolemap.file.CmdLine

  a. classpath:
    - LWI_NLS_LIB, previously initialized in setupCmdLine.sh
    - /www/<IAS_name>/lwi/lib/lwirolemap.jar
    - /www/<IAS_name>/lwi/runtime/core/rcp/eclipse/plugins/com.ibm.lwi.security.authengine_1.0.0.jar

  b. parameters:
    - **-roleDataDir**
    - /www/<IAS_name>/lwi/conf/admin
    - those passed into this lwiMapRole.sh script

Echoes SUCCESS if the java executed successfully

**lwimsg.sh**

IBM Internal Use Only

**lwiplugin.sh**
IBM Integrated Web Application Server for i

**Purpose**

Used to deploy and manage bundles in the IAS instance. The IAS instance must be running or this command will fail.

**Usage**

```
-lwiplugin.sh {–start | -stop | -undeploy | -status} <plugin_id_nbr | plugin_name>
-lwiplugin.sh –deploy <url_to_plugin>
-lwiplugin.sh –help
```

**Usage Notes**

- **-start** Indicates that the specified `plugin_id_nbr` should be started. The `plugin_id_nbr` is the number of the plug-in, which can be retrieved by running `lwiplugin –status`.

- **-stop** Indicates that the specified `plugin_id_nbr` should be stopped. The `plugin_id_nbr` is the number of the plug-in, which can be retrieved by running `lwiplugin –status`.

- **-deploy** Deploys a plug-in jar file. The `url_to_plugin` value must be the fully-qualified path to the jar file. See Section 0, lwiCreateExtension.sh to deploy plug-ins into an IAS instance when the instance runtime is not running.

- **-undeploy** Undeploys a plug-in. The `plugin_id_nbr` is the number of the plug-in, which can be retrieved by running `lwiplugin –status`.

- **-status** Displays the status of the running plug-ins. If a `plugin_id_nbr` is specified, the status for just that plug-in will be displayed. If not specified, then the status for all the active plug-ins will be displayed. The following are the possible status values that will display:
  - **INSTALLED** – the bundle/plugin is installed but is unable to successful start. There are several possible reasons including missing resources. For example, some required bundle can not be found. The detailed reason can often be seen when starting the bundle. Use the `lwiplugin.sh` script, specifying the “-start” parameter to try and start this bundle and see the issues. This is frequently used when deploying customized application.
  - **RESOLVED** – The bundle/plugin has successfully been deployed to the server and all necessary resources have been found. Bundle is just not yet running.
  - **Active** – The bundle/plugin is active and available for use.

- **plugin_name** The `plugin_name` is the bundle symbolic name assigned on startup by the runtime.

- **-help** Displays command usage.

**Command Logic**

1. executes setupCmdLine.sh

2. changes the current directory to `/www/<(IAS name)/lwi/runtime/core`
IBM Integrated Web Application Server for i

3. executes the class com.ibm.lwi.utils.cli.PluginUtils (what does this do?)
   a. classpath:
      i. LWI_NLS_LIB, previously initialized in setupCmdLine.sh
      ii. variables previously initialized in the setJars.sh script, previously called by setupCmdLine.sh (LWIUTILS_JAR, WEBCONTAINER_JAR, SHAREDVALUE_JAR)
   b. parameters: the arguments passed into this lwiplugin.sh script

4. echoes SUCCESS if the java completed successfully, otherwise exits with the java execution return code

IwiRemoveConfigFile.sh

Purpose
Removes a configuration file from the IAS instance. If the IAS instance is running, it will need to be restarted for the changes to take effect. Certain operating systems will not allow a file to be deleted if a process currently has it open. In these cases, it will be necessary to shut down the IAS instance before running the command.

Usage
IwiRemoveConfigFile.sh {-properties | -jar | -module | -jaasconfig | -help} config-file

Usage Notes
- **properties** Indicates that the configfile.properties file should be removed from the /www/<IAS_name>/lwi/conf/overrides directory of the IAS instance.
- **jar** Indicates that the configfile.jar file should be removed from the /www/<IAS_name>/lwi/lib directory of the IAS instance.
- **module** Indicates that the configfile should be removed from the /www/<IAS_name>/lwi/native directory of the IAS instance.
- **jaasconfig** Indicates that the configfile.config file should be removed from the /www/<IAS_name>/lwi/security/jaas/addtlconfig directory.
- **help** Displays command usage
- **configfile** absolute path to the configFile.

Command Logic
1. Executes setupCmdLine.sh
2. Executes the class com.ibm.lwi.config.RemoveConfigFile
   a. classpath:
      i. LWI_NLS_LIB, previously initialized in setupCmdLine.sh
      ii. /www/<IAS_name>/lwi/conf/configLWI.jar
   b. java arguments: -Dlwi.dir=/www/<IAS_name>/lwi
   c. parameters: the arguments passed into this lwiplugin.sh script
IBM Integrated Web Application Server for i

Echoes SUCCESS if the java completed successfully

IwiRemoveExtension.sh

**Purpose**
Remove a directory link created previously with the IwiCreateExtension command. The plug-ins/features in the directory will no longer be loaded when the IAS instance starts. If the IAS instance is active (running), it will need to be re-started for the plug-ins/features status to be updated.

**Usage**
IwiRemoveExtension.sh <pathToExtension> [-help]

**Usage Notes**
- **pathToExtension** is a link directory which was previously passed to the IwiCreateExtension command
- **-help** Displays command usage

**Usage Examples**
If the plug-ins are located in: /opt/IBM/xyz/eclipse/plugins, and had previously been deployed using the IwiCreateExtension command, then:

IwiRemoveExtension.sh /opt/IBM/xyz

**Command Logic**
1. Executes setupCmdLine.sh
2. Executes the class com.ibm.lwi.config.RemoveExtension
   a. classpath:
      i. LWI_NLS_LIB, previously initialized in setupCmdLine.sh
      ii. /www/<IAS_name>/lwi/conf/configLWI.jar
   b. java arguments: -Dlwi.dir=/www/<IAS_name>/lwi
   c. parameters: the arguments passed into this lwiplugin.sh script

Echoes SUCCESS if the java completed successfully, otherwise exits with the java execution return code

Iwirestart.sh

**Purpose**
Restarts the IAS instance. The Iwirestart command performs the following process:
- Stops the currently running IAS instance from the IAS instance directory where the Iwirestart command is being executed (/www/<IAS_name>/lwi/bin)
IBM Integrated Web Application Server for i

- Starts an IAS instance from the IAS instance directory where the `lwirestart` command is being executed (/www/<IAS_name>/lwi/bin). This lightweight runtime instance is started without displaying the OSGi console.

**Usage**
lwirestart.sh [-help | -?]

**Usage Notes**
- **-help or -?** Display command usage

**Command Logic**
1. executes setupCmdLine.sh
2. checks for arguments of `-?` or `-help`. If found, prints out the usage for the command
3. executes lwistop.sh
4. executes lwistatus.sh until it receives a non-zero return code
5. executes lwistart.sh with all the parameters passed into this command

**IwiSetJavaHome.sh**
(Changing; will need to be updated)

**Purpose**
Sets the JRE to be used by the IAS instance.

**Usage**
lwiSetJavaHome.sh <java_home_dir> [-help]

**Usage Notes**
- **java_home_dir** must be an absolute path to the JRE installation directory (where the /bin/java command is located)
- **-help** Displays command usage
- for System i, only /QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit is currently allowed
- Note that using this command to set the Java home, vs the export command, will not make the variable $JAVA_HOME available to subsequent QSH commands. If you try to execute `echo JAVA_HOME=$JAVA_HOME` after executing this command, it will display: `JAVA_HOME=`

**Coding Examples**
IBM Integrated Web Application Server for i

lwiSetJavaHome.sh /QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit

Command Logic

1. sets variable SKIP_JAVA_SETUP (to tell setupCmdLine to skip setting up the JVM) then executes setupCmdLine.sh

2. sets the variable JAVA_HOME to the value passed as the 1st parameter into this command (which should be the path of the Java home, i.e. /QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit)

3. echoes the JAVA_HOME that was set

4. echoes SUCCESS if the command was successful (return code of 0)

lwistart.sh

Purpose

Starts the IAS instance. This script is the direct way to start the server. The information in the i5Startup.Properties file is ignored by this command. The IAS server being started by this command is started using the User ID that is currently running this QSH environment. This will likely not cause any issues starting the server this time, but it should be noted that the could be issues later when starting the server using the STRTCPsvr command as that will start the server with the IAS runtime user. When an IAS server is started, several cache files are created and owned by the User ID that the server is running under. This means if using the default User ID QLWISVR, the server will not start if QLWISVR does not have authority to these files. It is recommended to not use this command for starting the server. If you absolutely do need to use this command, signon to the system with the IAS runtime User ID. This will ensure that the authorities are accessible later.

Note: This only starts the application server, not the associated HTTP (Apache) server.

Usage

lwistart.sh [-help]

Usage Notes

-help Displays command usage

Command Logic

1. executes setupCmdLine.sh

2. Adds execute permission to the launchlw i.sh script and executes the script with the parameters passed into this lwistart.sh script
Iwistarti5.sh

For IBM internal use only.

Iwistatus.sh

Purpose
Retrieves the current status of the IAS instance. Returns zero (0) for an ACTIVE instance state, and a non-zero for all other statuses. Note: this only returns the status of the application server, not the HTTP (Apache) server.

Usage

Iwistatus.sh [-fullversion | -help]

Usage Notes

- **-fullversion** displays the IAS version details. The result of using the – fullversion argument will be similar to the following: “lightweight runtime 7.1.1.0 IBM OS/400 PowerPC 0825B_53lwi711-20080711 MAX”
- Without the –fullversion argument, a simple ACTIVE or INACTIVE will be returned
- **-help** Displays command usage

Command Logic

1. executes setupCmdLine.sh
2. changes current directory to /www/<IAS_name>/lwi/runtime/core
3. checks for arguments of ‘-?’, ‘-help’ or ‘-h’. If found, prints out the usage for the command
4. initializes the ACTION variable. If the 1st parameter passed into the command is ‘-fullversion’, then sets ACTION to –fullversion, otherwise sets ACTION to the default of ‘-status’
5. executes the class com.ibm.lwi.utils.cli.StopStatusUtils (what does this do?)
   a. classpath:
      i. variables initialized in setJars.sh, which is called by setupCmdLine.sh (LWIUTILS_JAR, WEBCONTAINER_JAR, SHARDBUNDLE_JAR)
      ii. the LWI_NLS_LIB variable, initialized in setupCmdLine.sh
   b. parameters
      i. 1st parameter is the variable ACTION, previously initialized in this script
      ii. The parameters passed into this script
IBM Integrated Web Application Server for i

**lwistop.sh**

*Purpose*

Stops the IAS instance. Note: This does NOT stop the associated HTTP (Apache) instance, only the application server.

*Usage*

    lwistop.sh [-help]

*Usage Notes*

- **-help** Displays command usage

*Command Logic*

1. executes setupCmdLine.sh
2. changes current directory to /www/<IAS_name>/lwi/runtime/core
3. executes the class com.ibm.lwi.utils.cli.StopStatusUtils with a first parameter equal to -stop, followed by the parameters that were passed into this command.

**lwiupdatemgr.sh**

*Purpose*

Provides the command line interface to Eclipse Update Manager (EUM) functionality in IAS 7.1. The command provides the ability to manage features in the IAS runtime. Whenever the `lwiupdatemgr` command is used to change the status of features in the platform, the platform needs to be restarted to have these changes fully reflected in the platform.

*Usage*

- `lwiupdatemgr.sh -help`
- `lwiupdatemgr.sh --installFeatures [-featureId feature_id] [-version version] [-fromSite siteLocation] [-toSite siteLocation] [-configId configuration_id]`
- `lwiupdatemgr.sh --uninstallFeature -featureId feature_id -version version`
- `lwiupdatemgr.sh --enableFeature -featureId feature_id -version version [-configId configuration_id]`
- `lwiupdatemgr.sh --disableFeature -featureId feature_id -version version -configId configuration_id`
- `lwiupdatemgr.sh --updateFeatures [-featureId feature_id] [-version version] [-fromSite siteLocation] [-configId configuration_id]`
- `lwiupdatemgr.sh --listFeatures [-fromSite siteLocation]`  
- `lwiupdatemgr.sh --listConfig [-configId config_id]`
- `lwiupdatemgr.sh -rollback -configId configuration_id [-shutdown]`

*Usage Notes*
IBM Integrated Web Application Server for i

- **help** Displays command usage.
- **installFeatures** Indicates new features are to be installed and enabled.
  - **--featureId** Specifies the Id of the feature to be installed and enabled. If this parameter is not specified, all features in the specified site will be installed and enabled.
  - **--version** Specifies the version of the feature to be installed and enabled. This parameter is only valid when the **--featureId** parameter is specified.
  - **--fromSite** Specifies the local or remote site where the feature(s) will be installed from. If this parameter is not specified, the default site /www/<IAS_name>/lwi/install.images is used.
  - **--toSite** specifies the configured site where the feature(s) will be installed and enabled. If this parameter is not specified, the features are installed and enabled in the IAS’s default configured site, /www/<IAS_name>/lwi/apps/eclipse.
  - **--configId** Specifies the configuration identifier that can be used to rollback this modification if needed.
- **uninstallFeature** Indicates features to be uninstalled and disabled.
  - **--featureId** Specifies the Id of the feature to be disabled and uninstalled.
  - **--version** Specifies the version of the feature to be disabled and uninstalled.
- **enableFeature** Indicates an installed feature is to be enabled.
  - **--featureId** Specifies the Id of the feature to be enabled.
  - **--version** Specifies the version of the feature to be enabled.
  - **--configId** Specifies the configuration identifier that can be used to rollback this modification if needed.
- **disableFeature** Indicates an installed feature is to be disabled.
  - **--featureId** Specifies the Id of the feature to be disabled.
  - **--version** Specifies the version of the feature to be disabled.
  - **--configId** Specifies the configuration identifier that can be used to rollback this modification if needed.
- **updateFeatures** Indicates existing features are to be updated.
  - **--featureId** Specifies the Id of the feature to be updated. This parameter is only valid when the **--version** parameter is specified. If this parameter is not specified, then a search is done for updates for all enabled features in all configured sites.
  - **--version** Specifies the version of the feature update. This parameter is only valid when the **--featureId** parameter is specified.
  - **--fromSite** Limits the search for feature updates to just the specified local or remote site.
  - **--configId** Specifies the configuration identifier that can be used to rollback this modification if needed.

Once the list of features to update is determined, the following updates sites are searched for updates to these features:

- If the optional **--fromSite** operand is specified, this is the only update site that is checked for updates. If appropriate updates are found for features to be updated, then the updates are processed. Otherwise, processing stops and no updates are performed.
IBM Integrated Web Application Server for i

- If the optional \texttt{--fromSite} operand is NOT specified, then the update policy control file is first checked to determine if there are entries for the features to be updated. If entries exist for features to be updated, then the associated update sites are checked for updates for these features. If appropriate updates are found for these features, then the updates are processed from the update sites. If updates are not found for these features on the update sites, then no update processing is done for these features.

- If the optional \texttt{--fromSite} operand is not specified, and no entries exist in the update policy control file for features to be updated, then the update site specified in a feature's feature.xml file is searched for an update. If an appropriate update is found, then the update is processed. If no update is found on this update site, processing stops for the feature and no update is performed.

\begin{itemize}
  \item \textbf{-listFeatures} Indicates installed features are to be listed
    \begin{itemize}
      \item \texttt{-fromSite}
    \end{itemize}
  \item \textbf{-listConfig} List the modifications associated with configuration ids.
    \begin{itemize}
      \item \texttt{-configId}
    \end{itemize}
  \item \textbf{-rollback} Roll back modifications associated with the specified configuration id.
    \begin{itemize}
      \item \texttt{-configId}
      \item \texttt{--shutdown} Shuts down the IAS instance after the rollback operation completes.
    \end{itemize}
\end{itemize}

\textbf{Command Logic}

1. executes setupCmdLine.sh
2. changes the current directory to /www/<IAS_name>/lwi/runtime/core
3. executes the class com.ibm.lwi.utils.cli.FeatureUtils (what does this do?)
   \begin{itemize}
     \item classpath:
       \begin{itemize}
         \item LWI-NLS_LIB, previously initialized in setupCmdLine.sh
         \item Variables initialized in setJars.sh, which is called by setupCmdLine.sh
           (LWIUTILS_JAR, WEBCONTAINER_JAR, SHARDBUNDLE_JAR)
       \end{itemize}
     \end{itemize}
   d. parameters: the arguments passed into this lwiupdatemgr.sh script
4. exits with the java execution return code

\textbf{lwiupdatepolicy.sh}

\textbf{Purpose}

Modifies an update policy control file. The location of the update policy control file to be modified is defined by the system property org.eclipse.update.core/updatePolicyURL. The update policy control file contains feature prefixes and the site URL where features are to be updated from.

\textbf{Usage}

\begin{itemize}
  \item \texttt{lwiupdatepolicy.sh --help}
  \item \texttt{lwiupdatepolicy --addURLMapPattern -urlMapPattern urlMapPattern [-url siteURL]}
\end{itemize}
IBM Integrated Web Application Server for i

- **lwiupdatepolicy** --removeURLMapPattern -urlMapPattern urlMapPattern

**Usage Notes**

- **-help** Displays command usage.
- **-addURLMapPattern** Add an additional URL map pattern to the lwi_install_directory/update.images file (/www/<IAS_name>/lwi/update.images file)
- **-urlMapPattern** Specifies a string that represents the prefix of a feature ID (up to and including a complete feature ID). Examples:
  - com.ibm.lwi
  - com.ibm.usmi.console
  - com.ibm.isc
- **-url** Specifies the site URL that contains the feature updates. If this operand is not specified, then the default lightweight runtime update site is used (the lwi_install_directory/update.images file, or /www/<IAS_name>/lwi/update.images file). Examples:
  - file:../update.images/ (this is the default lightweight runtime update site)
  - c:\myupdatesite
  - /tmp/updates
  - http://my_update_server/updates
- **-removeURLMapPattern** Removes a URL map pattern from the lwi_install_directory/update.images file (/www/<IAS_name>/lwi/update.images file).

**Command Logic**

1. executes setupCmdLine.sh
2. changes the current directory to /www/<IAS_name>/lwi/runtime/core
3. executes the class com.ibm.lwi.utils.cli.UpdatePolicy (what does this do?)
   a. classpath:
      i. LWI_NLS_LIB, previously initialized in setupCmdLine.sh
      ii. Variables initialized in setJars.sh, which is called by setupCmdLine.sh (LWIUTILS_JAR, WEBCONTAINER_JAR, SHARED BUNDLE_JAR, LWIDEPLOY_JAR)
   b. parameters: the arguments passed into this lwiupdatepolicy.sh script
4. echoes SUCCESS if java execution completed successfully, otherwise exits with the java execution return code

**setJars.sh**

**Purpose**

Declares and initializes variables to point to appropriate jar files required for subsequent commands/scripts. The initialized variables all point to the www/<IAS_name>/lwi/runtime/core/rcp/eclipse/plugins directory. Examples of variable s that are set up (not all-inclusive): DB2_JAR, LWIUTILS_JAR, LWIDEPLOY_JAR, etc..
IBM Integrated Web Application Server for i

In normal practice, it should not be necessary to execute this command/script. Every other command in this list executes the setupCmdLine.sh script which, in turn, executes this setJars.sh script.

Usage

setjars.sh

setupCmdLine.sh

Purpose

Initializes the QSH environment to be able to execute the other commands/scripts shipped with the IAS product. This initialization includes declaring and initializing various variables to point to appropriate jar files and directories and validation of the Java runtime.

In normal practice, this script should not have to be called explicitly. Each of the other commands provided (with the exception of setJars.sh) already calls this script.

Usage

setupCmdLine.sh

Command Logic

1. Initializes various variables to be subsequently used
   a. BIN_DIR = present working directory (/www/<IAS instance>/lwi/bin)
   b. LWI_DIR = /www/<IAS instance>
   c. START_DIR = present working directory (/www/<IAS instance>/lwi/bin)
   d. CONF_DIR = /www/<IAS instance>/lwi/conf
   e. NATIVE_DIR = /www/<IAS instance>/lwi/native
   f. LIB_DIR = /www/<IAS instance>/lwi/lib
   g. SECURITY_DIR = /www/<IAS instance>/lwi/security
2. Executes setJars.sh
3. Initializes and exports the IAS NLS lib variable (includes the lwin1.jar, lwin2.jar and lwin3.jar files at /www/<IAS instance>/lwi/lib/)
4. If the $SKIP_JAVA_SETUP argument is NOT included with the command (typical), then the java initialization is executed:
   a. Verifies that the JAVA_HOME has been set up (see notes at the beginning of this section).
   b. Validates that the JAVA_HOME contains a Java executable (java command)
D. Appendix D Property/XML/log/MF Files

It is unreasonable to list all of the files that exist, or could exist, in an IAS instance. This section lists those that are most relevant (“the 20% that are used 80% of the time”).

IAS Instance (Server) Files

These files are found in the /www/<IAS_name> directory.

Note: it is important, with the exception of the webcontainer.properties file, that files in /www/<IAS_name>/lwi/conf NOT be changed. These files are symlinks and changing them will change the files to which they are linked. This could have unknown impacts on this, and other, IAS servers. Changes should be made, instead, in the /www/<IAS_name>/lwi/conf/overrides directory.

Following are examples of the directory structures typically seen for IAS instances.

Figure 106 - Root directory for IAS instance InstAppTst

© Copyright IBM Corporation, 2008. All Rights Reserved. All trademarks or registered trademarks mentioned herein are the property of their respective holders.
IBM Integrated Web Application Server for i

Figure 107 \lwi\ Directory for IAS instance InstAppTst

Figure 108 conf directory for IAS instance InstAppTst

Figure 109 conf/overrides directory for IAS instance InstAppTst
**Figure 110 logs directory for IAS instance InstAppTst**

**iasCreate<IAS_name>.properties**
Contains a list of the properties used to create the IAS instance. Included are the ports chosen (web and apache), the java version, the type of instance (profile), the job info (subsystem, job queue, routing data, userid, etc.), and whether SSL was enabled.

**webcontainer.properties**
Contains information relative to the web container and its relationship to the Apache server. Included are properties specifying the minimum and maximum number of threads to be started on which to listen for Apache requests, the maximum number of concurrent (persistent) connections that are allowed, and SSL enablement properties (if the container is SSL-enabled).

**config.properties**
The file is represented in the SPI by the class `com.ibm.lwi.admin.spi.server.ConfigPropertiesFile`. 

**Figure 111 - Config.Properties file**

**database.properties**

The file is represented in the SPI by the class com.ibm.lwi.admin.spi.server.DatabasePropertiesFile.

**logging.properties**

The file is represented in the SPI by the class com.ibm.lwi.admin.spi.server.LoggingPropertiesFile.

**i5javaopts.javaopt**

This file contains all the jvm startup options.

It is located at - /www/server_name/lwi/conf/overrides/i5javaopts.javaopt
IBM Integrated Web Application Server for i

Figure 112 - java options file

**Note:** during the process of server startup, all files that end in 'javaopt' will be processed. If create a copy of this file and make updates, be sure to change the file type or it will be processed.

**virtualhost.properties**

The file is represented in the SPI by the class `com.ibm.lwi.admin.spi.server.VirtualHostPropertiesFile`.

**version.properties**

Lists the build information for the IAS instance, including the version, build date and build level. Typically, this information would be of interest only to IBM Support, as part of any problem isolation effort. The file is represented in the SPI by the class `com.ibm.lwi.admin.spi.server.VersionPropertiesFile`.

**lwistdout.txt**

The system standard output file contains messages indicating whether the code running on the application server started and stopped successfully, as well as any other information written to the standard output stream of the application server. A new file is created every time the server is started.

**lwistderr.txt**

The system standard error file contains errors thrown by the application server and by the code running on the application server. It contains information written to the standard error stream of the application server. A new file is created every time the server is started. Includes java version as well as framework and command line argument information.
IBM Integrated Web Application Server for i

*jobname.txt*

The current name of the IAS instance is logged here in number/user/name format.

*preamble.txt*

Contains error and trace information from the code that create the environment and start the integrated Web application server. If the server is not starting correctly this is a good place to look for details.

*preamble.err.txt*

Contains error and trace information from the code that create the environment and start the integrated Web application server. If the server is not starting correctly this is a good place to look for details.

*trace-log-n.xml*

The trace-log-n.xml file contains errors thrown by the application server and the code running on the application server. It also contains trace information generated by the application and the application server.

This trace information is in XML format. The trace-log-n.xml files are archived every time the server is started and ended. Up to 5 archive files will be maintained.

*error-log-n.xml*

The error-log-n.xml file contains errors thrown by the application server and the code running on the application server. The error information is in XML format. The error-log-n.xml files are archived every time the server is started and ended. Up to 5 archive files will be maintained.

*lwistart.txt*

The lwistart.txt contains information about how application server is going to be launched, including the java command line used, and the job name the application server instance gets assigned.
HTTP (Apache) Server Files

These files are found in one of three directories (default):
- /www/<IAS_name>/conf,
- /www/<IAS_name>/htdocs,
- /www/<IAS_name>/logs

If you specified a different Apache server name than the default when you created your IAS instance, then the Apache server files would be located in one of the following three directories:
- /<Apache_name>/<IAS_name>/conf,
- /<Apache_name>/<IAS_name>/htdocs,
- /<Apache_name>/<IAS_name>/logs

Following are examples of the directory structure typically seen for Apache servers.

Figure 113 root directory for Apache instance InstAppTst

Figure 114 conf directory for Apache instance InstAppTst
IBM Integrated Web Application Server for i

![Figure 115 logs directory for Apache instance InstAppTst](image)

**httpd.conf**

This is the main configuration file for the HTTP Apache server. It is a “read-once” file, so any changes made to the file require an Apache restart. The key pieces of information contained in this file include the document root for the server, the port(s) on which it will listen, and the IP addresses to which it will listen. A couple key directives included in this file is a definition of any IAS servers associated with the Apache server and the location of the plugin file (lwi-plugin-cfg.xml) that acts as a bridge between the Apache server and the IAS instance.

**i5OSStartup.properties**

The file is represented in the SPI by the class com.ibm.lwi.admin.spi.server.StartupPropertiesFile.

**lwi-plugin-cfg.xml**

The file is represented in the SPI by the class com.ibm.lwi.admin.spi.server.ServerPluginXmlFile.

**plugin.log**

**error_log.Q1yyyymmdd00**

**maint.log**
Application Files

These files are found in the /www/<IAS_name>/lwi/apps directory and its subdirectories. The content depends on the installed application. This location will contain any directories unique to the application that were included in the war/jar that was installed. However, at a minimum, the following relevant directories and files will exist:

Following are examples of the directory structure typically seen for applications installed into an IAS server.

**Figure 116 List of applications for IAS instance InstAppTst**

**Figure 117 root directory for application IASDocumentCode in IAS instance InstAppTst**

**Figure 118 WEB-INF directory for application IASDocumentCode in IAS instance InstAppTst**
**IBM Integrated Web Application Server for i**

**Figure 119 META-INF directory for application IASDocumentCode in IAS instance InstAppTst**

### plugin.xml

**TBD** This file contains... The file is represented in the SPI by the class `com.ibm.lwi.admin.spi.server.ApplicationPluginXmlFile`.

If a bundle must be automatically started at platform startup, the following extension point must be added to the bundle’s plugin.xml file (by default, it is):

```xml
<extension point="com.ibm.rcp.lifecycle.application.startBundles">
    <application id="com.ibm.lwi.application.LWIApplication">
        <bundle id="com.ibm.mybundle" />
    </application>
</extension>
```

This must be accompanied by the following entry in the MANIFEST.MF file:

*Bundle-SymbolicName: com.ibm.mybundle ; singleton=true*

The "lazy (or manual) start" header can also be used to start web applications when they are first accessed. This eliminates the need to have web application plug-ins be activated at IAS startup time. The following extension point specifies a lazy start (autostart = false):

```xml
<extension point="com.ibm.pvc.webcontainer.application">
    <contextRoot>/TEST</contextRoot>
</extension>
```

In this latter case. The MANIFEST.MF file entry will look as follows (the singleton portion is dropped):

*Bundle-SymbolicName: com.ibm.mybundle*
MANIFEST.MF

Contains OSGi bundle information associated with the application. Some pieces of information include: bundle symbolic name, dependent bundles and Java import packages required by the application. The file is represented in the SPI by the class `com.ibm.lwi.admin.spi.server.ManifestFile`.

wab.properties

Contains the context root for the application, which is used in URLs to access the application. The file is represented in the SPI by the class `com.ibm.lwi.admin.spi.server.WabPropertiesFile`.

web.xml

This is the deployment descriptor for the application. A deployment descriptor contains configuration data that the run-time environment uses for an application. This information can be quite extensive and includes: mime types supported, welcome files and error pages and included servlets and their mappings,
E. Appendix E - Example Code

The coding examples in this appendix are included with this document. They can be accessed via the IAS product Web site.  http://www-03.ibm.com/systems/i/software/ias/

Javadoc for the code is also available at the same location; open up Test Code/JavaDoc/index.html. The JavaDoc for a class will provide details on how to execute the code.

The example code is of two types. Some of the code consists of standalone classes, with main(String[]) methods that can simply be called by the Java executable. The JavaDoc for each example explains how this is done in QShell. These classes perform some kind of action that does not produce a significant amount of output (i.e. create an IAS instance, install an application, etc.).

The other type of code example is that which generates considerable output. These are generally query examples that gather and print information. DB access examples would fit into this category, since DB access needs the context of a specific IAS server web container to be able to access the defined DataSource. That context can not be obtained in a standalone program. Examples of this second type execute as servlets, and are accessed via a browser. Basically, this means creating an IAS server and installing the IASTestServlets.war file (provided with the example code that accompanies this document) into the created server. This configuration activity is more easily done via the GUI, and would be a good place to start for those new to the technology. However, It could also be done programmatically, using the non-servlet coding examples described in the previous sections of this appendix. The IASTestServlets.war file also contains the source for the servlets, for your convenience.

Should you desire to enhance the code or the Javadoc for your own benefit, we would recommend the following practice:
1. Move the code to a location of your choosing. This will prevent the original code from being overlaid, should you refresh the code (download again) at some point in the future.
2. Make the changes you desire.
3. Re-generate the Javadoc, if you have changed any:
   a. TBD
   b. TBD

These coding examples were tested and verified as of the date of this document and under the following PTF group levels:
   o TBD

SPI Overview

Many of the coding examples (and programs you might create), use the same basic principles each time. The following classes, and their subclasses, are the keys to using the SPI.

IntegratedServerAdmin class

For overall management of an IAS server, at a high level, including creating, removing, starting and stopping, the IntegratedServerAdmin class is available. See the JavaDoc for additional capabilities of
this class. This is a convenience class for the programmer; under the covers, it is making use of the
server management and server classes discussed later in this section. It is most convenient to work at
this level, if it provides the solution you need. Its intent is to provide the “20% of the capability used
80% of the time”.

The constructor is used to obtain an instance of this class:

```
IntegratedServerAdmin admin = new IntegratedServerAdmin(serverProps);
```

Where `serverProps` is a Properties object that provides the necessary information for the class to be
able to do its work. That information will vary, depending on what task is to be performed. For in-
stance, to remove an IAS server, the Properties object only needs the name of the server to be cre-
ated and a boolean indicating whether the associated HTTP (Apache) server should also be removed.
Creating an IAS instance, on the other hand, requires quite a few properties to be initialized. The cod-
ing examples and the JavaDoc for the class are the best sources to use to determine the necessary
properties for any particular action.

**Server Management classes**

These classes are similar in purpose to the IntegratedServerAdmin class above. They provide capa-
bility for creating, removing and administering IAS server instances.

The IAS server management area is defined at a high level by two classes: ServerManagerFactory
and ServerManager. As you might expect, the ServerManagerFactory is used to obtain an instance of
ServerManager:  

```
ServerManager sm = ServerManagerFactory.getServerManager();
```

The ServerManager is actually an interface that begins a hierarchy:

- ServerManager (Interface)
- ServerDirector (implements ServerManager) – provides basic implementations of its imple-
  mented interface and adds capabilities that are not directly related to the IAS server, but to its
  environment. Examples include getting the platform object, or getting the user registry.
- i5OSServerDirector (extends ServerDirector) – provides capabilities that are unique to the par-
  ticular server; in this case, System i.

This is a hierarchy that is also common to the ApplicationManager and DatabaseManager classes be-
low. As is the case with each of these hierarchies, it is recommended to work at the highest level pos-
sible. Much of the time, that would mean working at the ServerManager level. However, the sub-
classes have additional methods and, depending on your requirements, you may have to work at the
lower levels to accomplish your objective. These lower methods, while providing additional capability
and flexibility, are also more complex, and require more of the developer.
Server classes

The server classes may seem different from the management classes in the SPI however, much of the difference is in the naming. The hierarchy in this area is:

- ManageableInstance (Interface)
- IasInstanceImpl (implements ManageableInstance) – Similar to the ServerDirector above, provides implementations of its implemented interface and addresses capabilities that are indirectly related to the server. It deals with topics such as setting java options, registry entries and/or the application and database managers.
- I5OSIasInstance (extends IasInstanceImpl) – provides implementations of methods that are unique to the given platform, in this case, System i.

Servers are retrieved in the following manner:

```java
ServerManager sm = ServerManagerFactory.getServerManager();
ManageableInstance mi = sm.getServer(<serverName>);
```

Where:

- `serverName` is a String specifying the name of the server to be retrieved

Again, similar to the server/application/database management classes, the hierarchy entails an interface, an implementer and an extender. The higher up in the hierarchy that you can work, the more convenient it is.

Application Management classes

These classes provide the interfaces for managing applications. By now, the class hierarchy structure should be familiar:

- ApplicationManager (Interface)
- ApplicationDirector (implements ApplicationManager) – provides implementations of its implemented interface, plus some additional platform-generic capabilities.
- ISOSApplicationDirector (extends ApplicationDirector) – provides platform-specific implementations of methods, where necessary. In this case, they would be System i implementations.

The ApplicationManager is retrieved as follows:

```java
ServerManager sm = ServerManagerFactory.getServerManager();
ManageableInstance ins = sm.getServer(serverName);
ApplicationManager mgr = ins.getApplicationManager();
```

Where:

- `serverName` is a String specifying the name of the server to be retrieved
Database Management classes

These classes provide the programmer with the interfaces required to manage database connections. The, by now familiar, hierarchy is:

- DatabaseManager (Interface)
- DatabaseDirector (implements DatabaseManager)
- I5OSDatabaseDirector (extends DatabaseDirector)

The DatabaseManager is retrieved as follows:

```
ServerManager sm = ServerManagerFactory.getServerManager();
ManageableInstance ins = sm.getServer(serverName);
DatabaseManager mgr = ins.getDatabaseManager();
```

Where:

- `serverName` is a String specifying the name of the server to be retrieved

Managing the Server

In addition to the following coding examples, there are a few other properties that can be set for a server to help manage the server. These are demonstrated in the myproperties.properties file provided with this document. It is a file that would be placed into the `/www/<ias server instance>/lwi/conf/overrides` directory and would be read at server startup time. There is no SPI interface to set or query these properties. See the properties file for more details.

Creating an IAS Instance

See the coding examples:

- Server_createInstance.java
- Server_createInstance_2.java

The difference between the two is that the ‘_2’ version uses a higher level interface (Integrated-ServerAdmin class) to accomplish the task. Coding for this is easier, and it is recommended that you use this version if you can, but it does not provide as much flexibility as the other version.

Removing an IAS Instance

See the coding examples:

- Server_removeInstance.java
- Server_removeInstance_2.java

The difference between the two is that the ‘_2’ version uses a higher level interface (Integrated-ServerAdmin class) to accomplish the task. Coding for this is easier, and it is recommended that you use this version if you can, but it does not provide as much flexibility as the other version.

Starting/Stopping Servers

See the coding examples:

- Server_startOrStopServer.java
- Server_startOrStopServer_2.java
The difference between the two is that the ‘_2’ version uses a higher level interface (Integrated-
ServerAdmin class) to accomplish the task. Coding for this is easier, and it is recommended that you
use this version if you can, but it does not provide as much flexibility as the other version.

Managing Applications

**Pre-install Validation**
See the coding example App_precheckApplication.java.

**Installing an Application**
See the coding example App_installApplication.java.

**Removing an Application**
See the coding example App_removeApplication.java

**Starting/Stopping an Application**
See the coding example App_startOrStopApplication.java.

Querying Application and/or Attributes
- status
- is duplicate application name

Managing Application and/or Attributes
- Setting/changing application properties
  - context root
  - autostart
- Adding an import package
- Adding a required (dependent) bundle

Database

**Adding a Database Connection**
This DB_addConnection coding example demonstrates how a DB connection would be added pro-
gramrnatically. A template of the required addConnection.properties file is also provided, and includes
descriptions of all the properties that can be managed when creating the connection. Of particular
note, especially for those with a WebSphere background, is the information and capability provided for
auto commit and Prepared Statement caching. The path to the addConnection.properties file is pro-
vided as an argument when the DB_addConnection Java class is executed. See the JavaDoc for this
class for more details and for instructions on setup and execution.

**Removing a Database Connection**
DB_removeConnection programmatically removes an existing database Connection Id from the speci-
fied server. The Connection ID to be removed and the IAS instance name on which the connection
exists are passed as arguments when the DB_removeConnection class is executed. See the JavaDoc
for this class for more details and for instructions on setup and execution.

**Validating a Database Connection**
ValidateDBConnectionServlet validates the connection id on several fronts:
that the DataSource can be accessed
- that the connection id exists on the specified server
- that a connection can be retrieved from the connection id pool
- that the connection can be used to access the Database (userid and pwd)
- that the schema provided on the connection is valid (if a valid SQL statement is provided on the input JSP)

If a valid SQL statement is not provided as input, to be used for the validation check, then the check will be done using database meta data. This will execute all the same validations as executing SQL, EXCEPT for the Schema validation.

This coding example is a bit different than the other DB examples provided. While only two pieces of information are required for input (the connection id name to be validated and the name of the IAS instance in which the connection id exists) and there is only one piece of response information (valid or not), this example is the only one that actually USES a connection. It therefore needs to execute within an IAS web container, so a servlet is used instead of a standalone Java program. See the JavaDoc for the ValidateDBConnectionServlet class for more details and for instructions on setup and execution.

Also note the exception handling in the coding example. This is an area where the design is not straightforward. In particular, some exceptional conditions are “hidden” and it is difficult to determine the root cause programmatically (without looking in the log files). This is a situation that is being corrected at the next SPI version, which will allow for “cleaner” code.

**Updating a Database Connection**

All of the same properties that were set when the Connection was created can also be subsequently updated. This DB_updateConnection coding example demonstrates those updates. Similar to adding a connection, a template of the required updateConnection.properties file is provided with this document and includes descriptions of all the properties that can be changed. Of particular note, especially for those with a WebSphere background, is the information and capability provided for auto commit and Prepared Statement caching. The path to the updateConnection.properties file is provided as an argument when the DB_updateConnection Java class is executed. See the JavaDoc for this class for more details and for instructions on setup and execution.

**Sample Servlets**

Some of the example classes generate considerable output, or output that needs to be specifically formatted. These classes are constructed as servlets and are accessed through a browser. Basically, this means creating an IAS server and installing the IASTestServlets.war file (provided with the example code that accompanies this document) into the created server. This configuration activity is more easily done via the GUI, and would be a good place to start for those new to the technology. However, it could also be done programmatically, using the non-servlet coding examples described in the previous sections of this appendix. The IASTestServlets.war file also contains the source for the servlets, for your convenience.

**Querying Platform**

- Type of platform (i, Linux, Windows, AIX, etc.)
- IAS product version
- IAS product install root
- Default java version
Querying Servers and/or Attributes

- List servers
- Getting IAS server instance properties:
  - Config properties
  - Startup properties
  - Version properties
  - Webcontainer properties
  - Java options
- Associated HTTP (Apache) server configuration
- Getting IAS product information
- Getting IAS version information
- List instance log files

Querying Application and/or Attributes

- list applications for server
- application properties
  - bundle name and symbolic name
  - classpath
  - context root
  - import packages
  - location
  - ports
  - is autostart
  - is .jar or .war file

Querying Database Connections

- List connections
- Connection exists
- List connection attributes

**ValidateDBConnectionServlet**

This servlet validates a database connection that you have created. It does this by accessing request-ing meta data using the connection you have configured. If successful, it will print a successful mes-sage. A failure will print out the exception received.

Prior to executing this example, you must have created at least one connection id that you will want to validate.

Use the following URL to execute the servlet:

```
http://<system>:<port>/IASTestServlets/ValidateDBConnectionServlet
```
Where:
- **system** is a domain name or IP address that provides access to the system on which the IAS server resides
- **port** is one of two ports:
  - the listening port of the HTTP (Apache) server associated with the IAS instance, or
  - the port assigned to the application when it was installed. This is the internal port of the IAS server, and bypasses the Apache server.

*PrintIasJvmPropertiesServlet*

This servlet will access the System (JVM) properties for the server in which this servlet is executing and print out the key-value pairs of those properties in the browser.

Use the following URL to execute the servlet:

```
http://<system>:<port>/IASTestServlets/PrintIasJvmPropertiesServlet
```

Where:
- **system** is a domain name or IP address that provides access to the system on which the IAS server resides
- **port** is one of two ports:
  - the listening port of the HTTP (Apache) server associated with the IAS instance, or
  - the port assigned to the application when it was installed. This is the internal port of the IAS server, and bypasses the Apache server.

*ListPlatformAttributesServlet*

For code that will access and display information associated with the platform on which an IAS instance is executing, see the example ListPlatformAttributesServlet.

Use the following URL to execute the servlet:

```
http://<system>:<port>/IASTestServlets/ListPlatformAttributesServlet
```

Where:
- **system** is a domain name or IP address that provides access to the system on which the IAS server resides
- **port** is one of two ports:
  - the listening port of the HTTP (Apache) server associated with the IAS instance, or
  - the port assigned to the application when it was installed. This is the internal port of the IAS server, and bypasses the Apache server.

*PCCustomerQueryServlet*

This servlet shows how to use the different database connection IDs that have been defined for the server instance to access 2 different sets of data from within the single application. Two connection IDs (can be local, remote, or a mix) will be used, each pointing to a different schema (library) on the same system. The same table (physical file) exists in both schemas, but the data in each of the tables is different. By using the same SQL statement (`SELECT * FROM QCUSTCDT ORDER BY CUSNUM`)
ASC), but executing them on the 2 different connections, different results are achieved and printed to
the browser.

Prior to executing this servlet, some configuration is necessary:

- The QCUSTCDT table in schema/library QIWS must be copied to a schema/library of your
  choice. Change a few of the entries in the copied table, so that you will be able to confirm that
  the different connection IDs did, in fact, access different tables.
- The DB connections must be configured for the server:
  - The DEFAULT connection, associated with the QIWS schema
  - The MYDBCONN connection, associated with the schema to where you copied the
    QCUSTCDT table.

Use the following URL to execute the servlet:

   http://<system>:<port>/IASTestServlets/PCCustomerQueryServlet

Where:

- **system** is a domain name or IP address that provides access to the system on which the IAS
  server resides
- **port** is one of two ports:
  - the listening port of the HTTP (Apache) server associated with the IAS instance, or
  - the port assigned to the application when it was installed. This is the internal port of the
    IAS server, and bypasses the Apache server.

**DBTimerServlet**

This servlet demonstrates the performance of database queries for 3 different access scenarios:

- Using the non-pooled IBM i (formerly i5/OS) native DB2 JDBC driver
- Using the non-pooled IBM Toolbox for Java JDBC driver
- Using the database connection pooling that is built into the IAS database connections

Using the SQL statement, "SELECT * FROM QIWS.QCUSTCDT", the servlet executes each scenario
ten (10) times and records the results, including the average access time for each scenario. The re-
results are printed in a table in the browser. Note: The library and file in the SQL statement should al-
ready exist on your system.

This particular servlet requires a valid user profile and password for the system on which the
IASTestServlets.war is installed. The place to provide this profile and password is in the
/www/<IAS_name>/lwi/apps/eclipse/plugins/IASTestServlets WEB-INF/web.xml file, as follows (re-
place MYPROFILE and MYPWD with valid values):

```xml
<init-param>
  <description/>
</init-param>
```

```xml
<init-param>
  <param-name>username</param-name>
  <param-value>MYPROFILE</param-value>
</init-param>
```

```xml
<init-param>
  <param-name>equals</param-name>
  <param-value>MYPWD</param-value>
</init-param>
```
<param-name>password</param-name>
<param-value>MYPWD</param-value>

It is necessary to stop and restart the application after making this change and prior to executing the servlet (it is not necessary to restart the server itself). If the profile is not provided, or an invalid/expired profile is provided, the following message will be displayed in the browser when trying to access the servlet:

**Error 500: Authorization failure on distributed database connection attempt.**

Use the following URL to execute the servlet:

`http://<system>:<port>/IASTestServlets/DBTimerServlet`

Where:
- **system** is a domain name or IP address that provides access to the system on which the IAS server resides
- **port** is one of two ports:
  - the listening port of the HTTP (Apache) server associated with the IAS instance, or
  - the port assigned to the application when it was installed. This is the internal port of the IAS server, and bypasses the Apache server.

**DBTimerServlet2**

This performs the same function as the DBTimerServlet, except that it uses a JSP to display the results. DBTimerServlet generates the HTML directly from the Java code. It is generally recommended to separate the user interface and the business logic by using a JSP.

The URL for executing this servlet is: `http://<system>:<port>/IASTestServlets/DBTimerServlet2`, otherwise see **DBTimerServlet** for more information.