PureWeb® STK 4.0

Server Administration Guide

About Calgary Scientific

Calgary Scientific Inc. is dedicated to providing advanced visualization, web enablement and mobility enhancement solutions to industries looking for secure access and use of their data or graphics intensive applications, while using their existing systems**.** Visit www.calgaryscientific.com for more information.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Calgary Scientific Inc. cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Calgary Scientific products.

Trademarks

Released by

Calgary Scientific Inc. www.calgaryscientific.com.

Document Version: PW4.0_ServerAdmin_Guide_07-2013_v1.000.00

Table of Contents

Preface

Welcome to the *PureWeb STK Server Administration Guide*, part of the PureWeb Software Transformation Kit (STK) documentation suite.

Intended Audience

This document is intended for software developers and system administrators who plan to install and configure the PureWeb server.

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to techpubs@calgaryscientific.com.

Contacting Calgary Scientific Support

Use one of the methods in the table below to contact Calgary Scientific support.

Web Site	E-Mail
support.getpureweb.com	support@getpureweb.com

Chapter

1 Introduction

The PureWeb STK (software transformation kit) is a development platform that enables the rapid transformation of enterprise software into cloud-ready, web and mobile applications.

PureWeb-enabled solutions are composed of a service application, one or several client applications, and a web server.

The PureWeb server is designed to support PureWeb-enabled applications developed using the PureWeb STK. Applications are installed into the server using its plugin architecture.

The server, which can be deployed on either Windows or Linux operating systems, leverages existing web server technology (Apache Tomcat) to broker communications between a service and its client applications. It serves as a session manager, fields connections from clients, as well as launches and load-balances service instances. The server manages collaborative sessions, allowing two or more users to interact with the same service using independent clients. It also coordinates client disconnection and server process termination, and provides a variety of server diagnostic and application information.

This document describes how to configure the PureWeb server. Some of the configuration topics covered include security (SSL), authentication, and cluster management. This document also describes how to add PureWeb-enabled solutions to the server after they have been developed using the PureWeb STK.

Note: This document does not provide instructions on how to install the server. This information can be found in the *PureWeb Installation Guide*.

Chapter

2 Server Overview

This chapter describes the procedures to start and stop the server, as well as to log in and log out.

It also provides an overview of the server's user interface and menu options.

Starting and Stopping the Server

You can start and stop the server either using the desktop icons or commands.

Using the Desktop Icons

- To start the service, double-click on the **Start PureWeb** desktop icon.
- To stop the service, double-click on the **Stop PureWeb** desktop icon.

Using Commands

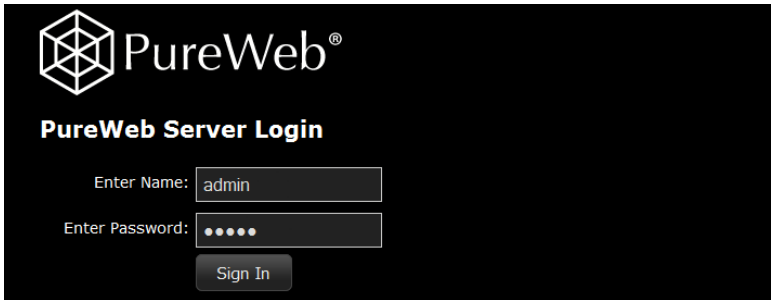
1. Open a console (command prompt) window.
2. Navigate to the following location (the default installation location is C:\CSI\PureWeb)
   `[Installation_Location]\Server\tomcat\bin`
3. Type one of the following commands:
   - To start the server: `startup.bat`
   - To stop the server: `shutdown.bat`
4. Close the server console window.

Logging In and Logging Out

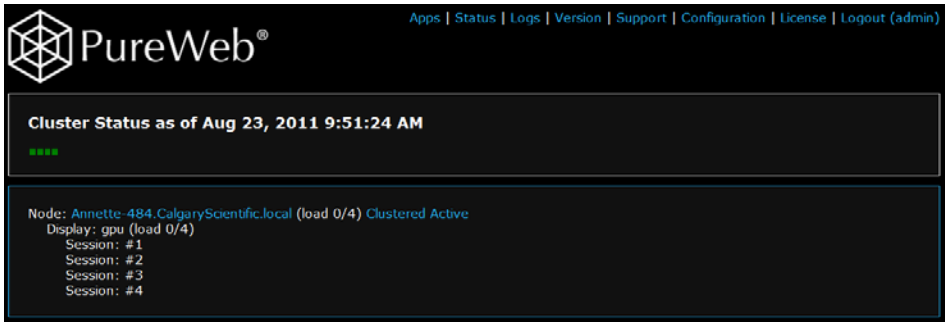
Use a web browser to access the PureWeb server as described below.

Logging In

1. Navigate to the server's URL; the default URL is http://localhost:8080.



2. Enter your login credentials. The default credentials use `admin` for both the name and password fields. To change this, see "Changing the Administrator Password" on page 15.
3. Click the **Sign In** button. This will display the server's landing page, as illustrated below.



For an overview of the server's interface and menu options, see "Navigating the Server's Interface" on page 10. Some of the options may not be available, depending on your user role; for more information, see "Security Roles" on page 27.

Logging Out

- From the server's user interface illustrated above, click on the **Logout** link in the top right-hand corner.

You will be logged out and the server's login page will be displayed.

Navigating the Server's Interface

The options available from the server's user interface are summarized in the table below. The last column specifies which security roles have access to each option.

Table 1: Server's Menu Options

Menu Option	Description	Accessibility
Apps	Provides access to the client applications. Depending on the user role used login, this page could appear as empty. For more information, see "Applications Page" on page 11.	Administrator Monitor User
Status	Displays the status page, which is used for cluster and node management. See "Cluster Management from the Status Page" on page 16.	Administrator Monitor
Logs	Displays a list of available log files (current and archived) recorded by the PureWeb server. See "Logs" on page 54.	Administrator Monitor
Version	Displays static information about the PureWeb server version. See "Version Page" on page 13	Administrator Monitor
Support	Displays server support information. The content on this page is customizable. See "Support Page" on page 13.	Administrator Monitor
License	Displays static information about the PureWeb server license. See "License Page" on page 14.	Administrator Monitor
Configuration	Provides functionality for server configuration. See "Configuration Page" on page 14.	Administrator
Logout	Logs the current user out of the PureWeb server.	Administrator Monitor User

Each of these options is described in more detail in the rest of this chapter.

Administrative URLs

You can navigate directly to any of the server's menu option by typing their URL in a browser. The URLs are as follows:

Applications page:
`http://localhost:8080/pureweb/server/links`

Status page:
`http://localhost:8080/pureweb/server/status`

Logs page:
`http://localhost:8080/pureweb/server/logs`

Version page (web):
`http://localhost:8080/pureweb/server/info`

Version page (raw text):
`http://localhost:8080/pureweb/server/info.txt`

Support page:
`http://localhost:8080/pureweb/server/support`

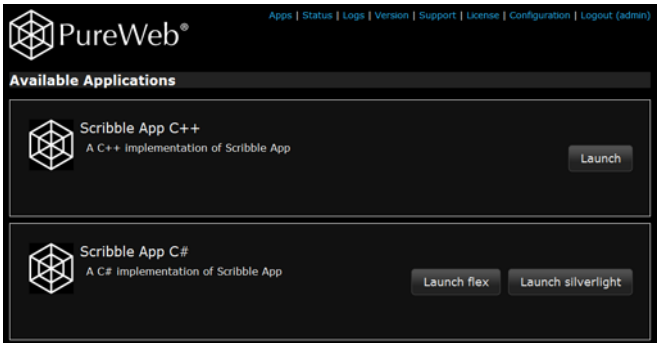
License page:
`http://localhost:8080/pureweb/server/license`

Configuration page:
`http://localhost:8080/pureweb/config/plugins`

Login page:
`http://localhost:8080/pureweb/server/login.jsp`

Applications Page

The Apps menu option lists the applications that are available on the server.



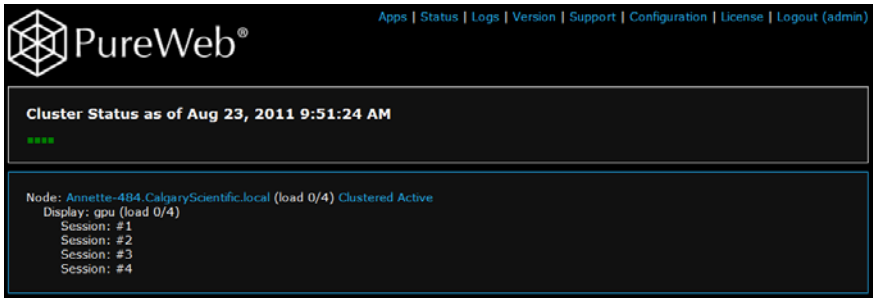
You can launch any of application by clicking on its **Launch** button.

When logged in as administrator or monitor, this page could be empty. This behaviour depends on whether these roles have been grated the necessary rights to see applications.

For detailed information on how to add applications to the server, see "Applications Management" on page 44.

Status Page

The server's Status page, which is the landing page first displayed when you log in, is used to manage clusters and nodes.

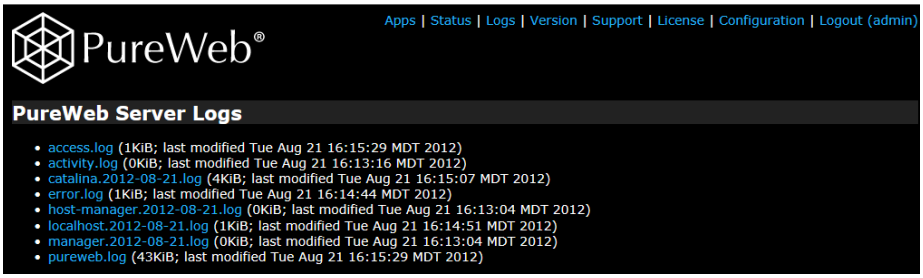


It contains a list of all the nodes in a cluster, all the display devices on each node, and all the configured sessions on each display device. Active sessions also display a list of all client connections.

For detailed information on this page, see "Cluster Management from the Status Page" on page 16.

Logs Page

The Logs menu option of the server's interface displays a list of log files.



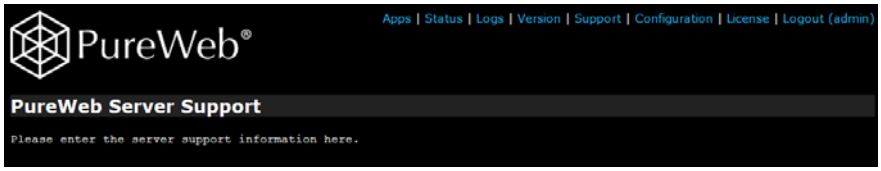
For a description of these files, as well as for log configuration and monitoring procedures, see "Logs" on page 54.

Version Page

The Version menu option displays build and version information about various aspects of the PureWeb server.

Support Page

The Support menu option displays information on who to contact to get support about the PureWeb server.



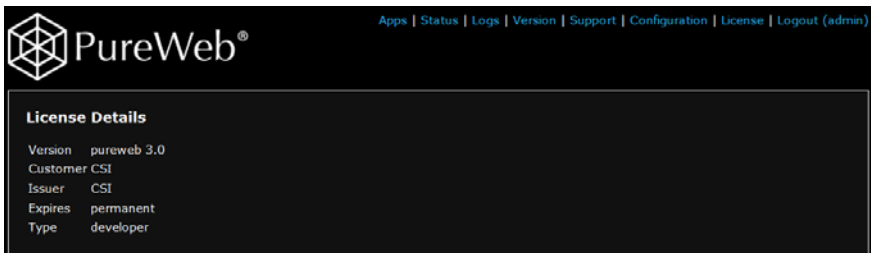
This page is especially useful when the server is deployed live at customer sites who run PureWeb-enabled applications.

The information on this page can be customized. The procedure is as follows:

1. Navigate to the PureWeb server's Configuration page by clicking on the **Configuration** link in the server's menu.
2. Scroll down to the bottom of the page, until you see the **Administration** section.
3. Click the **Change Server Support Information** link to open for editing.
4. Enter the custom support information.
5. Click the **Save** button to commit your changes.

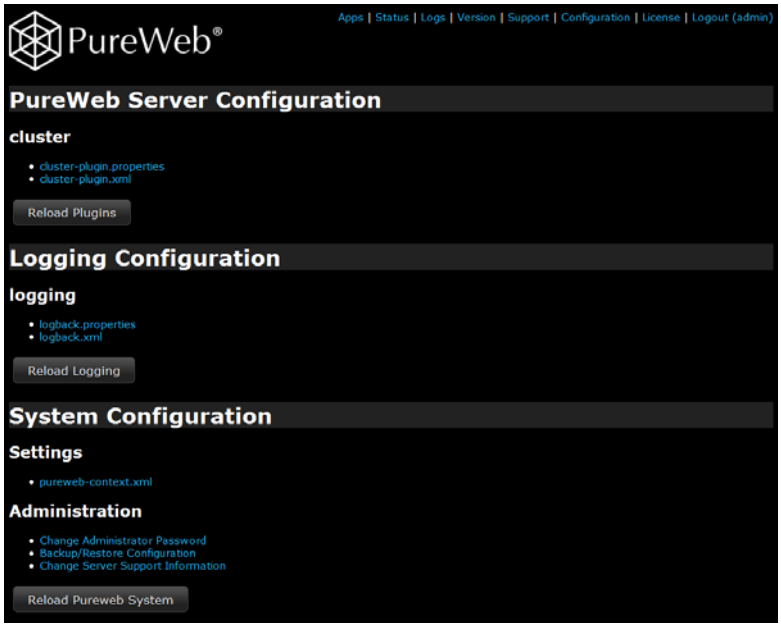
License Page

The License menu option displays license details, including version, type and expiration date.



Configuration Page

The Configuration menu option displays links to `.properties` and `.xml` files that contain server, logging and system configuration information.



It also contains tools to backup/restore the system configuration.

For more information, see "Configuration Page Overview" on page 22.

Changing the Administrator Password

After you have installed the PureWeb server, you should change the default administrator password.

1. Navigate to the PureWeb server's Configuration page by clicking on the **Configuration** link in the server's menu.
2. Scroll down to the bottom of the page, until you see the **Administration** section.
3. Click the **Change Administrator Password** link to open for editing.
4. Enter your old and new password information in the fields provided.

   New passwords must be at least 9 characters long.
5. Click the **Change Password** button.
6. Click the **Reload PureWeb System** button at the bottom of the Administration section on this page for the changes to take effect.

For information on how to change passwords for users who are not administrators, see "Changing a User's Password" on page 32.

Chapter

3 Cluster Management from the Status Page

The server's Status page, which is the landing page first displayed when you log in, is used to manage clusters and nodes.

Note: Server clustering and/or load balancing may not be necessary for your implementation of PureWeb; a single server may be sufficient.

The Status page contains information about each node in a cluster, the number of displays for each node, the maximum number of image sessions allowed per node, and more. Each element on this page is described in this chapter.

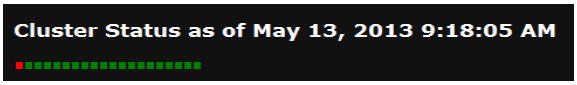
Status Page Features

This section provides a description of each of the elements found in the Status page.

Cluster Status Bar

The cluster status bar indicates how much of the cluster's total graphic display allocation is currently in use relative to overall availability.

The red indicates current usage, the green indicates current availability.



The status bar and its timestamp are not updated automatically at regular intervals. To update them, refresh the page in the browser.

Node Name

Each node in the Status page is identified by its name, an example of such a name would be `myserver.calgaryscientific.com`.

Node State Links

A cluster node can be either clustered, unclustered, and active or inactive. These states are represented by two links.

One link is used for the clustered/unclustered state. If currently clustered, the link will be blue and its label will read **Clustered**; if currently unclustered, the link will be red and its label will read **Unclustered**.

The other link is used for the active/inactive state. If a cluster is active, the button will be blue and its label will read **Active**; if it is deactivated, the link will be red and its label will read **Inactive**.

You will need to use these links to change a node's state from time to time, for example to gracefully shut it down or to test it before bringing it into service.

Shutting Down a Node

To gracefully shut down an active, clustered node, first render it inactive (click on the **Active** link to change the status). This will redirect any requests for new sessions to another active, clustered node. Once all existing sessions have completed, mark it as unclustered (click on the **Clustered** link to change its status) so that it will not receive any further requests. You can now shut down the node without affecting any existing sessions.

Note that changes made here are not persistent. When you restart the server, it uses the values in the following file:

```
[Installed_directory]\Server\conf\cluster-plugin.xml
```

Testing a Node

To allow a node to be tested before it is brought into service, first configure it as unclustered but active and then start it for testing. Once testing is complete, shut down the node, reconfigure it as clustered and active, and then restart it to bring it into full service. The table below summarizes the various possible node states.

Table 2: Cluster Node States

Status	Description
Clustered	Nodes that are participating in the cluster. They redirect requests for new sessions to the least busy node in the cluster.
Unclustered	Nodes that are *not* participating in the cluster. They do not redirect requests for new sessions to other nodes in the cluster.
Active	Nodes that can accept requests for new sessions. They accept these requests if they are among the least busy nodes in the cluster.
Inactive	Nodes that cannot accept requests for new sessions.

Display Details

Each display corresponds to an available software display used by the server node. You can think of a display as a collection of slots for available PureWeb sessions. For each of these, the Status page provides the information describes in the sections below.

Display Name

Each display in a node is identified by a unique name. Examples of display names would be unix:0.0 and unix:0.1 on Linux, or gpu.0 or gpu.1 on Windows.

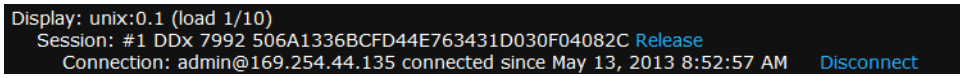
Display Capacity and Number of Active Sessions

This is the information in parenthesis after the display name. For example (2/20) means that there are two active imaging sessions out of 20; this second number, 20, indicates the maximum capacity for the display.

Sessions list

Below the display name is a sessions list, as illustrated below.

If users are connected to an image session, additional lines will appear below that session in the list, as illustrated below.



These lines contain links for disconnecting users and terminating active rendering sessions:

- Click the **Release** link to forcibly release the session. This disconnects all connected clients and shuts down the associated process.
- Click the **Disconnect** link to disconnect a single user interacting in a session. The session is automatically released when the last client is disconnected.

Cluster Configuration

You can edit the `cluster-plugin.properties` file (accessible from the server's Configuration menu option) to modify a number of properties that affect hosting and clustering, including:

- display properties, which are used to create a list of display devices available to the server
- process properties, which are used to create rules for interaction between the server and a service, such as response timeout and shutdown timeout
- client properties, which are used to create rules for interaction between the server and a client application, such as activity timeout and process limits
- cluster properties, which are used to group several servers together. Each cluster is defined by a specific cluster address and port.

The complete list and description of these properties can be found in the section "Hosting and Clustering Properties" on page 38.

Load Balancing

The PureWeb solution has a very strong server affinity requirement. All requests need to go to a single server and you need to direct collaboration sessions to the same server.

Traffic can be routed in two different ways, either using unique hostnames, or using a load balancer.

Using Unique Hostnames

You can give each node of the server farm a unique hostname. (Do not have a single hostname point to a server farm.)

In this case, your setup should have the following characteristics:

- Create a local hostname to each node of the farm.
- Create a corresponding hostname on the load balancer for each node. For simplicity, the load balancer's host name can be the same as the node's hostname.
- Configure the load balancer to direct all traffic from the same host to the same node in the farm. A secondary node may be set up as a backup. If the primary server goes down, bring the secondary node online.
- If the node in the farm responds to a different URL than the load balancer, configure the load balancer as a reverse proxy.

Using a Load Balancer

You can use a load balancer to route all traffic, without making the servers directly accessible. Your setup should have the following characteristics:

- Each server must have a unique hostname on the network that is internal to the load balancer.
- The load balancer must be configured to route requests to one of the servers. Sticky sessions, or session affinity, must be used with the JSESSIONID cookie to ensure that all traffic for one session is directed to the same server.
- The load balancer must be configured not to modify the headers of the request as they are forwarded to the server. It must also not modify the response. This requirement is to preserve hostnames in the request so URLs can be generated that are valid for the client.
- The load balancer must be configured to add a header containing an identifier for the server that will handle the request when before it is passed on. This is used to embed a host identifier in collaboration URLs.
- The load balancer must be configured to inspect GET requests for /pureweb/share URLs and use the embedded host identification information to route the request to the correct server.

Reverse Proxies

This section discusses the use of reverse proxies when the proxy is pointing to a single instance of PureWeb.

Note: Not all of the reverse proxies tested work in the standard connection mode (full duplex) provided by Microsoft Silverlight.

The alternative is to use the High Performance Polling connection mode instead.

All server status pages depend on relative URLs that point to the base directory. If the reverse proxy modifies the path location of PureWeb, you must translate all links in status pages.

In order for PureWeb to work through reverse proxies, first perform an analysis on the body of certain responses and translate the URLs to the proxied URL, then translate the response to a POST request for the following URLs:

- `/pureweb/app`
- `/pureweb/app/*/*/*`
- `/pureweb/app/*/*/*/*`
- `/pureweb/share`

You must parse the body of these responses to find links to the instance URLs. It is enough to look for `http://serverinstancehost:8080` for the host machine that is serving as a server instance.

All of the above URLs are of exact length. Do not translate any subdirectories below these URLs.

You only need to translate responses to POST requests, but you can translate others without problems.

Monitoring the Status Page

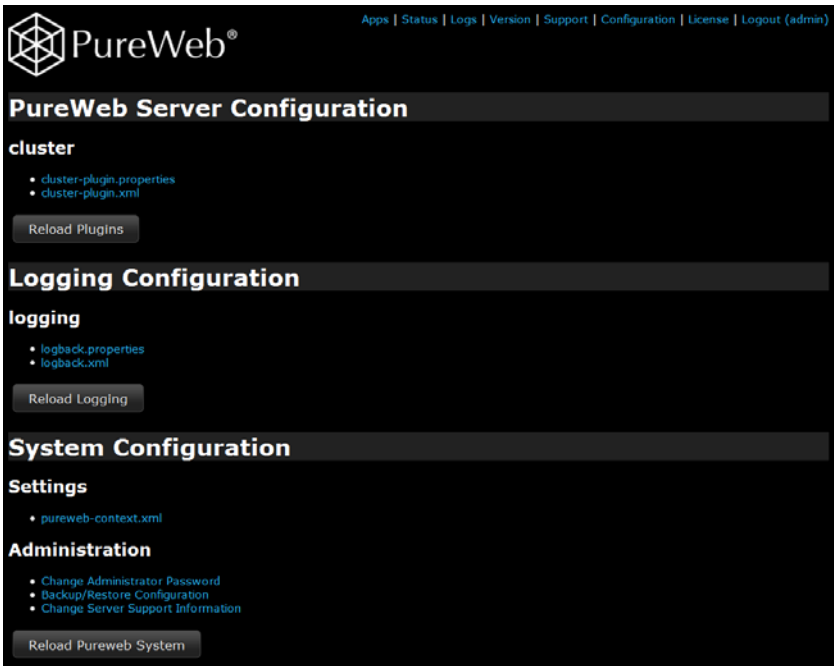
You can set up a monitoring tool such as Nagios (www.nagios.org) to send an alarm in the event that the home page returns a 404 error or it times out.

Chapter

4 Configuration Page Overview

This chapter gives a high-level overview of the server's Configuration page and provides links to the various configuration procedures described in this *Server Administrator's Guide*, as a quick reference.

It also describes the hierarchy of the configuration files in the installed directory, and how to backup and restore a PureWeb server's configuration information.



Configuration Page Features

The PureWeb server's Configuration page, available to users with administrator-level credentials, contains a list of configuration links, divided in three sections: at the top, general server configuration, followed by logging configuration, and finally system configuration.

Not all configuration files are accessible from this page. If a file is not on that page, it must be edited manually by opening it in a text editor from the server's installed directory. Whenever this is the case, instructions in this document provide the full path to the file.

Applications are added to the server using its plugin architecture. Which application (plugin) in installed will vary from implementation to implementation, and therefore the list of files listed on the Configuration page will also vary accordingly.

For information on how to add an application plugin, see "Applications Management" on page 44.

File Editing

The name of each file in the Configuration page is a link that opens the content of the file in an editor page within your browser.

The editor page has a simple interface where you can type in your changes, and a **Cancel** and a **Save** button at the bottom of the page.

After you edit a file and save your changes, the system returns to the Configuration page.

Note: Any configuration file that has pending changes will display `reload required`, indicating that changes have not been applied. To apply the changes, click the Reload button at the bottom of the section where the file is located within the server's Configuration page (for example, if you edited a plugin configuration file, click the **Reload Plugins** button).

Configuration Tasks

There are several configuration procedures described in the various chapters of this *Server Administrator's Guide* that involve editing files, either from the Configuration page, or directly in a text editor. Below is a list of these tasks for easy reference.

- Customizing the information in the support page (page 13)
- Changing the administrator password (page 15)
- Adding an LDAP authentication provider (page 29)
- Managing users; this includes adding or removing users, as well as changing their password or security role (page 31)
- Controlling access to applications using security roles (Page 28)
- Configuring the default response to unauthorized access (page 28)
- Configuring an HTTP/HTTPS connection (page 35)
- Configuring clusters and nodes; the properties in the cluster configuration files define the configuration of display devices, as well as interactions between the server and the service, and between the server and the client (page 37)
- Displaying links to applications within the server (page 51)
- Configuring the log file retention period (page 56)

Configuration File Hierarchy

The PureWeb server is implemented as a standard Java Web Application, using common Java components including the Apache Tomcat 7.0 web container (tomcat.apache.org) and various libraries from the Spring Framework (www.springsource.org).

The hierarchy of configuration files is defined by these technologies and the underlying plugin architecture defined by the server. Installation of optional plugins will add specific .xml and .properties files to the list of configuration files in the `conf` directory.

The table below lists the most common configuration files. The ellipsis (...) at the beginning of the path stands for the PureWeb server's installed directory; the default installed location is `C:\CSI\PureWeb`.

Table 3: Configuration File Hierarchy

Configuration File and Path	Description
...\webapp\WEB-INF\web.xml	Standard Java Servlet Specification 2.4 Web Application definition loaded by Apache Tomcat. Defines the PureWeb server web application.
...\webapp\WEB-INF\pureweb-context.xml	Standard Spring Framework Root Web Application Context loaded by the web application defined in `web.xml`. The PureWeb context defines the authentication facilities used by all components within the PureWeb server.
...\webapp\WEB-INF\dispatcher-servlet.xml	Standard Spring Framework Servlet Dispatcher loaded by the web application defined in `web.xml`. Defines common facilities used by various components of the PureWeb server.
...\conf\security-config.xml	Additional security configuration allowing various plugins to define their own security requirements using Java annotations.
...\conf\pureweb.xml	Common logging, error handling and configuration management facilities provided by the PureWeb server.
...\conf\cluster-plugin.xml	Clustered application hosting facilities.
...\conf\cluster-plugin.properties	Basic configuration properties referenced by the `cluster-plugin.xml` file above.

Configuration Backup and Restore

The server's Configuration page offers functionality that makes it easy to back up and restore your configuration information.

Backup Procedure

This procedure assumes that you are logged in and have navigated to the server's configuration page:

1. Scroll down the page until you see the **System Configuration** section.
2. Under Administration, click on the link labeled **Backup/Restore Configuration**. This will display the Configuration Backup page.
3. Click the **Save New Backup** button.

Your backup will appear under the **Save New Backup** button, in the form of a radio button labeled with the backup timestamp. As you create new backups, they will also appear in that section, in chronological order, with the most recent one at the bottom of the list.

The backup file itself is a .zip file saved at the following location:
`[Installed_directory]\pureweb\Server\conf\backups`

Restore Procedure

This procedure assumes that you have already created a backup as described above, and that you are logged in and have navigated to the server's Configuration page.

1. Scroll down the page until you see the **System Configuration** section.
2. Under Administration, click on the link labeled **Backup/Restore Configuration**.
3. Select the backup to restore from by clicking its radio button.
4. Click the **Restore Selected Backup** button.

Other Functionality

From the Configuration Backups page, you can use the export and import features to simplify the task of configuring a multi-node PureWeb cluster.

Export a selected backup to your client machine and save it there to establish common configuration on one server, then use the import option to restore on several other PureWeb servers.

The Configuration Backups page also has a **Delete Selected Backup** option, which deletes the selected backup permanently.

Chapter

5 Authentication and User Management

This chapter provides the necessary procedures to configure login authentication and authorization, in particular it describes how to set up LDAP for PureWeb.

This chapter also provides user management procedures, such as adding or deleting users, and changing their role or password.

To complete the procedures in this chapter, you must log in as an administrator.

Authentication

PureWeb uses Spring Security role-based authorization to control login authentication and authorization. Detailed information on this model can be found at the following location:

http://static.springsource.org/spring-security/site/docs/3.1.x/reference/springsecurity.html

Any authorization plugin that conforms to the Spring 3.1.x security model will work with PureWeb, as long as it grants applicable authority roles as described below.

Security Roles

The roles defined by the PureWeb server are as follows:

- **Administrator**: This role has access to all the server's menu options. It is the role required, for instance, to configure the plugins or to cancel sessions that are in progress. In Spring, it is defined as ROLE_PUREWEB_SERVER_ADMIN.
- **User**: This role is used for normal users of PureWeb applications, for instance it can launch client applications. In Spring, it is defined as ROLE_PUREWEB_USER.

- **Monitor**: This role is used for monitoring the server; it does not have access to the configuration pages, and cannot launch applications. In Spring, it is defined as ROLE_PUREWEB_SERVER_MONITOR.
- **Collaborator**: This role is used internally by PureWeb to define permissions for collaborators who join a PureWeb session. In Spring, it is defined as ROLE_PUREWEB_COLLABORATOR.

Authentication Configuration File

The PureWeb security information is in the `pureweb-context.xml` file.

To access this file within the server's user interface, navigate to the **Configuration** page and scroll down to the **System Configuration** section.

The file can also be edited directly in a text editor. It is located in the installed PureWeb directory:
`[installed_directory]\Server\webapp\WEB-INF\pureweb-context.xml`

Below are some configuration procedures that can be performed in this file.

Although it is also possible to configure LDAP using the context file, it is easier to create a plugin. For more information, see "LDAP Configuration" on page 29.

Controlling Access to Applications

The security policies allow the PureWeb server to restrict access to pages based on a user's role.

1. Navigate to the `pureweb-context.xml` file as described above.
2. Edit the values of the `intercept-url pattern` and `access` elements. For example:
   ```
   <s:intercept-url pattern="/pureweb/app/**"
   access="hasRole('ROLE_PUREWEB_USER')"/>
   ```
3. Click **Save** to commit the changes.
4. Close the file.

Changing the Default Response to Unauthorized Access

If an unauthorized user attempts to connect to a protected page, the server can respond in one of two ways:

- Basic authentication: the server will report an HTTP 401 Unauthorized error to unauthenticated clients. Standard web browsers will display an authentication dialog requesting the username and password.
- Form-based authentication: the server will redirect unauthenticated clients to an authentication page to request the username and password.

Form-based authentication is recommended and is selected by default; you can change this however; for example, if you are also using SSL, basic authentication may be sufficient.

To change the default response to unauthorized access:

1. Navigate to the `pureweb-context.xml` file as described above.
2. Comment out the form-based authentication section (lines 6 to 11) and remove comments for the basic authentication section (lines 13 and 16):.

```
1   <s:http auto-config="true" realm="PureWeb"
    createSession="always">
2   <s:request-cache ref="requestCache"/>
3   <!-- do not copy session attributes on authentication (in
    particular Client instances) -->
4   <s:session-management session-fixation-protection=
    "newSession"/>
5   <!-- form-based authentication -->
6   <s:form-login login-processing-url="/login"
7     login-page="/pureweb/server/login.jsp"
8     authentication-failure-ref="authenticationFailureHandler"
9     default-target-url="/pureweb/server/status"
10    always-use-default-target="false"/>
11  <s:logout logout-url="/logout"
    logout-success-url="/pureweb/server/logout.jsp"/>
12  <!-- basic authentication -->
13  <!--
14  <s:http-basic/>
15  <s:logout logout-url="/logout"/>
16  -->
```

LDAP Configuration

Spring Security includes a built-in LDAP provider, but this provider does not have the ability to map LDAP groups to custom roles. To offer this functionality, PureWeb has its own LDAP provider, `MappedLdapAuthenticationProvider`, which is otherwise similar to the one included with Spring Security.

The simplest way to use this provider in your application is through a PureWeb plugin. Create a new file called `ldap-plugin.xml` and place it at the following location:

`[installed_directory]\Server\conf`

Add the provider bean, illustrated on next page, to this file.

More information about managing plugins in the PureWeb server can be found in the chapter “Applications Management” on page 44.

`MappedLdapAuthenticationProvider` bean:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:s="http://www.springframework.org/schema/security"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
    http://www.springframework.org/schema/security
    http://www.springframework.org/schema/security/spring-security-3.1.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">

  <bean id="mappedLdapAuthenticationProvider"
  class="pureweb.util.MappedLdapAuthenticationProvider">
    <property name="serverUrl" value="ldap://host:389"/>
    <property name="serverUsername" value="manager-user"/>
    <property name="serverPassword" value="manager-password"/>
    <property name="userSearchBase"
    value="OU=...,OU=...,CN=...,DC=...,DC=..."/>
    <property name="userSearchFilter" value="(name={0})"/>
    <property name="groupSearchBase"
    value="OU=...,OU=...,CN=...,DC=...,DC=..."/>
    <property name="groupSearchFilter" value="(member={0})"/>
    <property name="roleMap">
      <map>
       <entry key="ldapRole1" value="PUREWEB_SERVER_ADMIN"/>
       <entry key="ldapRole2" value="PUREWEB_SERVER_MONITOR"/>
       <entry key="ldapRole3" value="PUREWEB_USER"/>
      </map>
    </property>
  </bean>
</beans>
```

The parameters need to match the LDAP parameters used by your directory. The table below provides a brief description of each parameter.

Table 4: Authentication Provider Bean Properties

Property	Value
serverUrl	The address of your LDAP or database authentication server.
serverUsername	The user name used to connect to the authentication server.
serverPassword	The password used to connect to the authentication server.

Table 4: Authentication Provider Bean Properties (Continued)

Property	Value
userSearchBase	The node in your tree to search for users.
userSearchFilter	The field in the user element to use as the user's name.
groupSearchBase	The node in your tree to search for groups.
groupSearchFilter	The field in the group element to find the user's name to see if they are in that group.
roleMap	A mapping of existing LDAP or JDBC roles to their PureWeb equivalent to provide site-specific configuration without requiring changes to existing LDAP directories or database roles.

Managing Users

This section provides the procedures for adding, removing or disabling users, and for changing their password or their role.

Generating an Encoded Password

The PureWeb server expects encoded passwords. Below are the procedures for generating such passwords, for Windows and for Linux servers.

Windows Servers

Use an online encoding tool that uses a SHA1 algorithm to produce an encoded password in a base64 format. For example:

http://www.webutils.pl/SHA1_Calculator

http://quickhash.com/

Linux Servers

Use the following command to generate encoded passwords:

```
$ echo -n "new-password" | openssl dgst -sha1 -binary | openssl
base64/v2lBCUbc9MKfPPQR9JBOIo8EbM=
```

Adding Users

1. Generate an encoded password for the user, as described in the procedure above.
2. Navigate to the following file and open it in a text editor:
   `[installed_directory]\Server\webapp\WEB-INF\user-service.properties`
3. Scroll down the file until you see a line in the following format:
   `[username]=[encodedpassword],[roles],enabled`
   Copy and paste that line and edit it using the information applicable to the user you are adding, for example:
   `janedoe=0DPiKuNIrrVmD8IUCuw1hQxNqZc=,ROLE_PUREWEB_USER,enabled`
4. Click **Save** to commit the changes, then close the file.
5. Restart the server for the changes to take effect.

Removing or Disabling Users

1. Navigate to the following file and open it in a text editor:
   `[installed_directory]\Server\webapp\WEB-INF\user-service.properties`
2. Scroll down the file until you see the line for the user you want to remove or edit, for example:
   `janedoe=0DPiKuNIrrVmD8IUCuw1hQxNqZc=,ROLE_PUREWEB_USER,enabled`
3. Do one of the following:
   - To remove the user, delete the line entirely.
   - To temporarily disable the user, replace the `enabled` parameter at the end with the value `disabled`.
4. Click **Save** to commit the changes, then close the file.
5. Restart the server for the changes to take effect.

Changing a User's Password

1. Generate an encoded password for the user, as described in the procedure above.
2. Navigate to the following file and open it in a text editor:
   `[installed_directory]\Server\webapp\WEB-INF\user-service.properties`
3. Scroll down the file until you see the line for the user whose password you want to change, for example:
   `janedoe=0DPiKuNIrrVmD8IUCuw1hQxNqZc=,ROLE_PUREWEB_USER,enabled`
4. Overwrite the existing password with the new one.

5. Click **Save** to commit the changes, then close the file.
6. Restart the server for the changes to take effect.

Note: The password for the administrative user can also be changed by following the steps provided in “Changing the Administrator Password” on page 15.

Changing a User’s Role

1. Navigate to the following file and open it in a text editor:
   `[installed_directory]\Server\webapp\WEB-INF\user-service.properties`
2. Scroll down the file until you see the line for the user whose role you want to change, for example:
   `janedoe=0DPiKuNIrrVmD8IUCuw1hQxNqZc=,ROLE_PUREWEB_USER,enabled`
3. Overwrite the existing role or add a new one (a user may have more than one role). See the section “Security Roles” on page 27 for a a list and description of the supported roles.
4. Click **Save** to commit the changes, then close the file.
5. Restart the server for the changes to take effect

Chapter

6 Secure Socket Layer (SSL)

This chapter is optional and used only for those sites that use Secure Socket Layer (SSL) connections to access the PureWeb server (some sites implement their own private security).

To complete the procedures described in this chapter, you must log in using administrator-level credentials.

Installing an SSL Certificate

SSL certificates are required for the PureWeb server to support HTTPS connections from PureWeb clients. A certificate must be acquired from a recognized certificate authority, or generated and configured in the Apache Tomcat container, before HTTPS connections are enabled.

SSL certificates may be requested from many different internet providers, all of which will follow a process similar to the one described in this section.

In this example, we use keytool, a key and certificate management utility. Keytool is included in the Java Development Kit (JDK).

1. Open a command prompt and navigate to the following directory:
   `[installed_directory]\tomcat\conf\`
2. Generate a certificate:
   ```
   keytool -genkey -alias tomcat -keyalg RSA -keystore
   tomcat.keystore
   ```
   The common name must be the fully qualified domain name of the host to which the certificate is applied.
3. Generate a certificate signing request (CSR):
   ```
   keytool -certreq -keyalg RSA -alias tomcat -file tomcat.csr
   -keystore tomcat.keystore
   ```

4. Submit the (CSR) to the certificate authority (CA). Generally, this will involve copying and pasting the CSR generated above into an online enrollment form and possibly indicating that the server software is Apache Tomcat.
5. Review the signed certificate that you receive from the certificate authority. This may be in the form of a zip file containing the signed certificate along with other certificates to form the certificate chain. For example:

```
keytool -import -alias cross -keystore tomcat.keystore
-trustcacerts -file gd_cross_intermediate.crt
keytool -import -alias intermed -keystore tomcat.keystore
-trustcacerts -file gd_intermediate.crt
keytool -import -alias tomcat -keystore tomcat.keystore
-trustcacerts -file tomcat
```

   Depending on which certificate provider you used, the files that you receive may be slightly different from those above. Regardless, you will need to import all of the files that you receive.
6. Import the signed certificate into a keystore.
7. Configure the Apache Tomcat container to use the signed certificate from the keystore.

Configuring an HTTP/HTTPS Connection

Before completing the procedure in this section, if you are enabling an HTTPS connection, ensure that you have a server certificate, as described above.

1. Navigate to the following file and open it in a text editor:
   `[installed_directory]\tomcat\conf\server.xml file`
2. To disable an HTTP connector: comment out lines 2-8 in the code shown on next page.
3. To enable an HTTPS connector:
   a. Remove the comments on lines 18 and 37.
   b. Ensure that the `keystoreFile`, `keystorePass` values on line 26 are set appropriately. If you created your certificate using the procedure in the first part of this chapter, you must modify the `keystoreType` value on this line as well.
   c. Optionally, restrict the set of available ciphers to those that are cryptographically strong by including the ciphers configuration on lines 39-45.
4. Save the file before closing.
5. Restart the server for the changes to take effect.

The default port for SSL configuration is port 8443. If port 8443 is blocked, configure a port of your choice instead by replacing line 19 as follows:

```
<Connector port="[chosen_port]" address="0.0.0.0"
```

Use the chosen port when accessing the server's administrative pages. For example: `https://localhost:443/pureweb/server/login.jsp`

The procedure in this section involves editing the following lines of code:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector port="8080" address="0.0.0.0"
  maxHttpHeaderSize="8192" emptySessionPath="false"
  maxKeepAliveRequests="-1"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" redirectPort="8443" acceptCount="100"
  connectionTimeout="600000" disableUploadTimeout="true"
  useBodyEncodingForURI="true" URIEncoding="UTF-8" />
<!-- Define a non-SSL HTTP/1.1 Connector on port 4502 for
Microsoft Silverlight socket connections -->
<Connector port="4502" address="0.0.0.0"
  maxHttpHeaderSize="8192" emptySessionPath="false"
  maxKeepAliveRequests="-1"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" redirectPort="8443" acceptCount="100"
  connectionTimeout="600000" disableUploadTimeout="true"
  useBodyEncodingForURI="true" URIEncoding="UTF-8" />
<!-- Define a SSL HTTP/1.1 Connector on port 8443 -->
<!--
<Connector port="8443" address="0.0.0.0" SSLEnable="true"
  maxHttpHeaderSize="8192" emptySessionPath=="false"
  maxKeepAliveRequests="-1"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" disableUploadTimeout="true"
  acceptCount="100" scheme="https" secure="true"
  clientAuth="false" sslProtocol="TLS"
  keystoreFile="conf/tomcat.keystore" keystorePass="password"
  keystoreType="PKCS12"
  <!-- keystoreType is optional, try removing it from the line
  above if the keystore is not working-->
  useBodyEncodingForURI="true" URIEncoding="UTF-8"
  ciphers="SSL_RSA_WITH_RC4_128_MD5,
    SSL_RSA_WITH_RC4_128_SHA,
    TLS_RSA_WITH_AES_128_CBC_SHA,
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
    TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
    SSL_RSA_WITH_3DES_EDE_CBC_SHA,
    SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
    SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA"/>
-->
<!-- enable specific (non-weak) ciphers -->
<!-- ciphers="SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
    TLS_RSA_WITH_AES_128_CBC_SHA,
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
    TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
    SSL_RSA_WITH_3DES_EDE_CBC_SHA,
    SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
    SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA" -->
```

Chapter

7 Clusters Configuration

This chapter describes some of the configuration tasks required to manage clusters and nodes. It also includes tables describing all the hosting and clustering properties available from the configuration files.

Configuring a Cluster

This procedure configures multiple PureWeb servers together into a cluster by editing the `cluster-plugin.properties` file.

1. Log into the server and click on the **Configuration** link in the menu at the top to open the Configuration page.
2. Scroll down the page until you see the **cluster** section.
3. Click on the `cluster-plugin.properties` file link to open it in the browser.
4. Edit the properties in this file as appropriate. See “Hosting and Clustering Properties” on page 38 for a description of these properties.
5. Click the **Save** button; you will be returned to the server’s Configuration page.
6. Click on the **Reload Plugins** button under the section PureWeb Server Configuration for the changes to take effect.

   Any plugins that have pending configuration changes will display reload required indicating that changes have not been applied.

New connections may be redirected to cluster nodes from which status updates have not been received within the timeout. The node selection algorithm should exclude these nodes.

Hosting and Clustering Properties

The properties in the `cluster-plugin.properties` file, described in this section, are used by the lower level `cluster-plugin.xml` configuration file to control specific features provided by the server.

Display Properties

Use these properties to create a list of display devices available to the server.

Table 5: cluster-plugin.properties Display Properties

Name	Description	Format	Windows Example

display.list	Defines the list of display devices available for use by this server. Each entry specifies the maximum number of client sessions allowed on a specific display device.	[max_sessions]@[display_device]
		where:
		• [max_sessions] is a positive integer (1..N)
		• [display_device] is the name of the associated display device.
		Entries for multiple display devices must be separated by spaces.
			Display token form = gnu or gpu.n
			where n is the GPU instance when using Quadro or ATI graphic processing units.
			Example:
			`display.list=4@gpu.0 4@gpu.1`
			Permits 4 users to share the first Quadro or ATI GPU and 4 users to share the second one.
			`display.list=4@gpu`
			Permits 4 users to share the available GPU resources on the system when a Quadro or ATI GPU is not being used.

Process Properties

Use these properties to create rules for application and server interactions.

Table 6: cluster-plugin.properties Process Properties

Name	Description
process.response.timeout	Defines the maximum number of seconds that an application process has to respond to a request from the server. If the application fails to respond within this timeout period, the server will consider it to be unresponsive and kill it. The default is 30 seconds.
process.shutdown.time-out	Defines the maximum number of seconds that an application process has to cleanly shutdown when requested by the server. If the application fails to shutdown within this timeout period, the server will consider it to be unresponsive and kill it. The default is 30 seconds.
process.cleanup.timeout	Defines the number of seconds between checks for inactive processes. Inactive processes will be released back to the cluster by the cleanup process. The default is 5 seconds.

Client Properties

Use these properties to create rules for client and server interactions.

Table 7: cluster-plugin.properties Client Properties

Name	Description
client.activity.timeout	Defines the maximum number of seconds that a client can consume a session with no client activity. If there is no activity from a client within this timeout the application process will be terminated and the session released to be available to other users. The default is 30 seconds.
user.process.limit	Defines the maximum number of rendering processes that each user of the system may have. Requests for rendering processes above this limit will be rejected. A value of 0 may be used to specify no process limit. The default value is 1.
silverlight.socket.policy.enabled	Set to true to enable the Microsoft Silverlight policy server or false to disable it. This is enabled by default.
silverlight.socket.policy.address	Defines the address that the xmlsocket policy server will listen on for permitting Microsoft Silverlight clients to connect using a raw (high-performance) socket.

Table 7: cluster-plugin.properties Client Properties (Continued)

Name	Description
silverlight.socket.policy.port	Defines the port that the xmlsocket policy server will listen on for permitting Microsoft Silverlight clients to connect using a raw (high-performance) socket. The Windows default is 943. This will require port redirection from 943->8943 (see tools/iptables.sh).
silverlight.socket.policy.hosts	Defines the domain and port patterns that Microsoft Silverlight clients will be permitted to connect to using a raw (high-performance) socket. Connections are only permitted on ports in the 4502 - 4534 range. The default is *:4502 (all domains on port 4502).

Cluster Properties

Use these properties to group several servers together into a cluster defined by a specific cluster address and port.

Table 8: cluster-plugin.properties Cluster Properties

Name	Description
cluster.enabled	Controls whether clustering is enabled. All enabled nodes with the same cluster address and port will form a cluster and balance client load across the cluster.
cluster.interface	Defines the multicast interface which will be bound to for all multicast communications. Leaving blank works, but with multi-NIC systems the bound interface will be indeterminate.
cluster.address	Defines the multicast address shared by all nodes in a cluster. Valid addresses must be in the range 224.0.0.0 through 239.255.255.255. Nodes will only form a cluster if they have the same cluster address and port.
cluster.port	Defines the multicast port shared by all nodes in a cluster. Valid ports must be in the range 0 through 65535 and should avoid the privileged ports with values below 1024. Nodes will only form a cluster if they have the same cluster address and port.
cluster.status.broadcast	Defines the number of seconds between status message broadcasts to the cluster. All nodes in a cluster send out periodic status updates notifying the other cluster nodes of their existence and current load conditions. These updates are broadcast whenever status changes occur on a node in addition to the frequency defined by this value. The standard value is 25 seconds.

Table 8: cluster-plugin.properties Cluster Properties (Continued)

Name	Description
cluster.status.timeout	Defines the number of seconds that a status message from another node is to be considered valid. The source node is marked as unresponsive if no status update is received by this timeout. This value should be set slightly higher than the status broadcast timeout to ensure that status updates are received and processed before a node is marked as unresponsive. The standard value is 30 seconds.
cluster.response.timeout	Defines the length of time that the PureWeb server should wait for responses from the service. Note that this property is used even if clustering is disabled. We recommend that you keep this value small, but never less than three seconds. When debugging increase the value to prevent the middle tier from closing the communication channels on a processed that has been stepped into with a debugger. Example: `cluster.response.timeout=10`
cleanupTimeout	Defines the amount of time to wait between periodic checks for inactive or abandoned client sessions. The default is 5 seconds. All nodes that are configured with the same cluster.address and cluster.port cooperate within a cluster and redirect requests as required to ensure that the load is distributed evenly.
application.connection.address	Defines the address for socket based service connections.
application.connection.port	Defines the port for socket based service connections.

Node Properties

The server will attempt to determine its hostname automatically, but occasionally there are system or network configuration problems that prevent this from being possible. In these cases, the hostname can be configured explicitly.

Table 9: cluster-plugin.xml Node Properties

Name	Description
hostname	Defines the hostname of this cluster node. This value is optional and will default to the system's hostname. The specified hostname must be resolvable via either the local hosts file or the configured DNS system.
clustered	Defines whether this node is participating in the cluster defined by the cluster address and port properties. Clustered nodes may redirect requests to other nodes to ensure an even load distribution across the cluster. Unclustered nodes will not redirect requests to other nodes and may report Service Unavailable errors if they have no available sessions.
active	Defines whether this node is active and will accept new connections or not. Active nodes will accept new connections if they have available sessions. Inactive nodes will not accept new connections even if they have available sessions.

Making a Service Available Across Domains

This procedure controls the domain access of clients. It is recommended for those using a cluster configuration.

1. Log into the server and click on the **Configuration** link in the menu at the top to open the Configuration page.
2. Scroll down the page until you see the **System Configuration** section.
3. Click the `clientaccesspolicy.xml` file link to display its content.

4. Edit the file by entering a domain name for the `domain uri` properties, lines 6 and 7 in the code snippet below.

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <access-policy>
3     <cross-domain-access>
4       <policy>
5         <allow-from http-request-headers="*" http-methods="*">
6           <domain uri="http://*" />
7           <domain uri="https://*" />
8         </allow-from>
9         <grant-to>
10          <resource path="/" include-subpaths="true"/>
11        </grant-to>
12      </policy>
13    </cross-domain-access>
14  </access-policy>
```

5. Click the **Save** button; you will be returned to the server's Configuration page.
6. Still from the Configuration page, open the `crossdomain.xml` file for editing.
7. Edit the file by entering a domain name for the `allow-access-from` domain property, line 5 in the code snippet below.

```
1   <?xml version="1.0"?>
2   <!DOCTYPE cross-domain-policy SYSTEM
    "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
3   <cross-domain-policy>
4     <site-control permitted-cross-domain-policies="master-only"/>
5     <allow-access-from domain="*"/>
6   </cross-domain-policy>
```

8. Click the **Save** button.
9. Click the **Reload PureWeb System** button under the System Configuration section for the changes to take effect.

Chapter

8 Applications Management

This chapter describes how to add an application and register it on the PureWeb server using a plugin XML file, and how to configure this application by creating a `.properties` file.

Configuration covers, in particular, adding resources and protecting them by limiting their access, allocating graphics processing units (GPUs), and providing a link to the application in the server's Applications page.

This chapter also provides a summary of all the files that may be added to the server's installed directory when an application is created, and therefore must be removed when the application is uninstalled.

Creating an Application Plugin XML File

The PureWeb server is designed to support PureWeb-enabled applications developed using the PureWeb STK. Applications are installed into the server using its plugin architecture.

The first step is to create the plugin XML file, as described in this section.

To facilitate this task, below is the code from a sample file that you can use as a model.

This sample code is then followed by tables which describes the properties within the various sections of the XML file.

Sample Plugin XML File

```
1   <?xml version="1.0" encoding="UTF-8"?>
2
3   <beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
4   import resource="security-config.xml"/>
5
6     <bean id="handlerMapping"
7       class="org.springframework.web.servlet.mvc.annotation.DefaultA
        nnotationHandlerMapping">
8       <property name="useDefaultSuffixPattern" value="false"/>
9       <property name="detectHandlersInAncestorContexts"
        value="false"/>
10    </bean>
11
12    <bean class="pureweb.servlet.PluginPropertyLoader"/>
13
14    <bean id="licenseManager"
      class="pureweb.process.DefaultLicenseManager"/>
15
16    <bean id="<application name>ProcessFactory"
      class="pureweb.process.ProcessFactory">
17      <property name="applicationRegistry"
        ref="applicationRegistry"/>
18      <property name="licenseManager" ref="licenseManager"/>
19      <property name="application" value="<application name>"/>
20      <property name="description" value="The Application"/>
21      <property name="directory" value="${<application
        name>.home}"/>
22      <property name="executable" value="${<application
        name>.executable}"/>
23      <property name="available" value="true"/>
24      <property name="defaultProcess" value="${<application
        name>.startup}"/>
25      <property name="environment">
26        <map>
27          <entry key="HOME" value="${<application name>.home}"/>
28        </map>
29      </property>

    //continued on next page
```

```
30      <property name="options">
31        <map>
32          <entry key="--<application name>" value="true"/>
33        </map>
34      </property>
35      <property name="arguments">
36        <list>
37          <value>/tmp</value>
38          <value>/var</value>
39        </list>
40      </property>
41    </bean>
42
43    <bean class="pureweb.cluster.SupportedClients"
      scope="prototype">
44      <property name="supportedClients" ref="supportedClientsMap"/>
45      <property name="application" value="example"/>
46      <property name="clientMap">
47        <map>
48          <entry key="silverlight" value="example.xap"/>
49          <entry key="ios"
            value="http://itunes.apple.com/us/app/example/id398740007?
            mt=8&uo=4"/>
50          <entry key="android"
            value="market://details?id=com.yourcompany.example"/>
51        </map>
52      </property>
53    </bean>
54  </beans>
```

The `PluginPropertyLoader` (line 12) loads application-specific configuration properties. If none are required, this line can be omitted.

Application startup information is specified in the `ProcessFactory` configuration (lines 16 to 41); see table “Process Factory Properties” on page 47.

Application client information is specified in the `SupportedClients` configuration (lines 46 to 58); see table “Supported Clients Configuration” on page 49.

Table 10: Process Factory Properties

Name	Description	Mandatory?
applicationRegistry	Defines the application registry that the process factory will register. If none is defined, the process factory will not be available for use.	Yes
licenseManager	Defines the license manager responsible for licensing of the specified application. The default license manager allows unrestricted access to the application. Third-party application plugins can define their own license manager to handle custom licensing requirements.	Yes
application	Defines the ID of the application that identifies the corresponding executable. This should be unique across all applications defined on the PureWeb server, or the server will error on startup and the plugin will not load.	Yes
description	Defines the name of the application that will be displayed to users.	Yes
directory	Defines the directory from which the application will be started. Applications that need to be started in a particular directory must specify an appropriate value in this property. This property is optional for applications that do not need to be started in a specific directory. If not specified, the working directory will be the PureWeb server's bin folder.	No
executable	Defines the full path to the application's main executable program. This program will be started with the specified environment, options and command line arguments when a request for the application is received.	Yes
available	Controls the availability of the specified application. This can be used so that the specified application is not listed among available applications and cannot be requested by clients.	No

Table 10: Process Factory Properties

Name	Description	Mandatory?
defaultProcess	Controls startup of the specified application. One application may be configured as the default process and the PureWeb server will eagerly start instances of this process in all available user sessions to avoid delays associated with process startup. Only one default process may be defined, otherwise only the first will be activated and subsequent process factory definitions will be ignored. A value of “true” indicates that this is the default process.	No

environment	Defines additional environment variables required by the application executable.
	Do not add a DISPLAY environment variable in this property.
	This property expects a map of key-value pairs, for example:
	`<property name="environment"><map>`
	`<entry key="LD_LIBRARY_PATH"`
	`value="."/>`
	`</map>`
	`</property>`
		No

options	Defines the command line options that the application must be started with.
	This property expects a map of key-value pairs, for example:
	`<property name="options">`
	`<map>`
	`<entry key="-c" value="server"/>`
	`</map>`
	`</property>`
		No

arguments	Defines the command line arguments that the application must be started with, in the order that they must appear.
	This property expects a list of values, for example:
	`<list>`
	`<value>arg1</value>`
	`<value>arg2</value>`
	`</list>`
		No

Table 11: Supported Clients Configuration

Name	Description	Mandatory?
supportedClients	Defines the client map that this application will register with. If no client map is defined the application will not be available for use.	Yes
application	Defines the ID of the application that identifies the corresponding client.	Yes
clientMap	Defines the name of the client file (relative to the webapp folder) or a link to the client (for mobile clients) for each supported platform. At least one of “silverlight”, “flex”, “ios”, “android” or “html5” must be specified. For clients of type of “android” or “ios”, the value specified is the market URL that the device will use to retrieve the client software when joining a collaborative session for the first time. Note: a link to an iOS client on the App Store may be generated using Apple's Link Maker (http://itunes.apple.com/linkmaker).	Yes

Adding the Plugin XML File to the Server

Once you have created the application’s plugin.xml file, follow the instructions below to add it to the server and register it.

1. Save the plugin xml file, using the following naming convention:
   `[application name]-plugin.xml`
2. Place the saved file at the following location:
   `[Installed_directory]\Server\conf`
3. Make the newly installed application available using one of the following two methods:
   - Reload the plugins (log into the server, navigate to the Configuration page, and click the **Reload Plugins** button at the bottom of the PureWeb Server Configuration section.
   - Restart the PureWeb server.

Creating a Plugin Properties File

Once you have created the properties file for your application as described below, the file will appear in the PureWeb server's Configuration page, and you will be able to edit the properties using that page.

1. Create a file in a text editor and enter your PureWeb-enabled application's properties using the following model:

   ```
   [application name].home=/home/[application name]
   [application name].executable=/home/[application
   name]/bin/[application name]>
   ```

2. Save the properties file, using the following naming convention:

   ```
   [application name]-plugin.properties
   ```

3. Place the saved file at the following location:

   ```
   [Installed_directory]\Server\conf
   ```

Managing Application-Specific Resources

The PureWeb-enabled application that you are adding to the server may have specific resources such as graphics.

This section describes how to add these resources to the server and protect them by limiting their access.

Adding the Resources

To add application-specific resources to the server, you must create a directory for them using the following path and folder naming convention:

```
[Installed_directory]\webapp\[application name]\
```

For example:

```
PureWeb\Server\conf\webapp\MyApp\
```

Place the resources in that folder.

Protecting the Resources

You can protect the application-specific resources using one of two methods:

- Editing the `security-config.xml` file
- Making resources private

Making resources private is the preferred approach, as it allows applications to define their own security constraints without needing to make changes to low-level configuration files.

Both options are described below.

Making Resources Private

1. Navigate to the following location:
   `[Installed_directory]\webapp\WEB-INF\views\`
2. Create a subdirectory.
3. Add the resources for your application to this directory.
4. Provide Controller implementations that make these resources available to users with the correct roles using the security annotations provided by the Spring Framework.

Editing the security-config.xml File

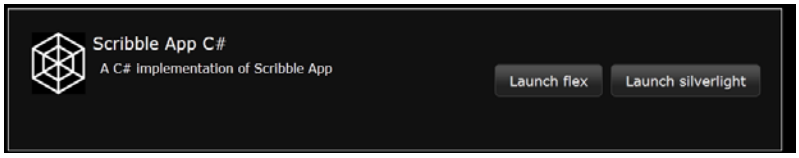
1. Navigate to the following file and open it in a text editor:
   `Installed_directory]\Server\conf\security-config.xml`
2. Edit the file by including this line:
   ```
   <intercept-url pattern="/<application name>/**"
   access="ROLE_PUREWEB_USER"/>
   ```
   This secures all of the resources using a simple pattern that recursively matches your resource directory; in this example access to the resources is limited to users who log in using a “user” level security role.
3. Save the file to commit you changes.

Displaying Links to the Application

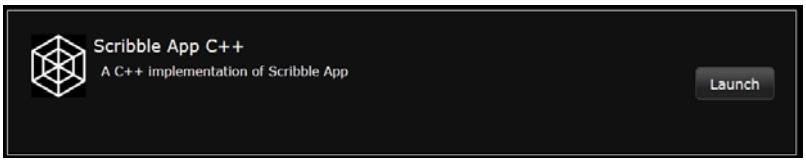
The application will not appear in the list of available applications on the server until you add the necessary links.

You can choose whether or not to provide an optional path; the link will look different based on this choice.

Link that includes the optional path



Link that does not include the options path:



To add the link:

1. Log in the server with administrator-level credentials.
2. Navigate to the Configuration page.
3. Scroll down the page until you see the plugin file for the application for which you want to add links (`[application name]--plugin.xml`).
4. Click on that filename link to open the file for editing and navigate to the section shown below.

```
1   <!-- Adds the link under the Apps section of the PureWeb Web
    Application -->
2   <bean class="pureweb.process.PluginLink">
3     <property name="registry" ref="pluginLinkRegistry"/>
4     <property name="supportedClients" ref="supportedClients"/>
5     <property name="name" value="Scribble App Advanced C#"/>
6     <property name="description" value="A C# implementation of
      Scribble App Advanced"/>
7     <!-- Optional. If specified the links generated will be as
      below. -->
8     <!-- Comment out the next line to exclude this option. -->
9     <property name="path"
      value="/pureweb/view?name=ScribbleAppAdvanced&amp;client=silverl
      ight"/>
10    <property name="image" value="/themes/pureweb/default.png"/>
11    <property name="role" value="ROLE_PUREWEB_USER"/>
12  </bean>
13
14  <bean class="pureweb.cluster.SupportedClients" scope="prototype"
    id="supportedClients">
15    <property name="supportedClients" ref="supportedClientsMap"/>
16    <property name="application" value="ScribbleAppAdvanced"/>
17    <property name="clientMap">
18      <map>
19        <entry key="silverlight" value="ScribbleAppAdvanced.xap"/>
20        <entry key="flex" value="ScribbleClientAdvanced.swf"/>
21        <entry key="ios" value=""/>
22        <entry key="android" value=""/>
23      </map>
24    </property>
25  </bean
```

5. Edit the file as follows:
   - To show multiple client options, comment out the `path` property in the `PluginLink` bean (line 9). When this is done, the various clients defined in the clientMap of the SupportedClients bean will be displayed.
   - To show only a single option, ensure that the path property is not commented out, and that the client parameter in the path points to the desired client.
6. Save the file to commit your changes.

The table below provides a description of the properties in the `PluginLink` bean.

For a description of the properties in the `supportedClients` bean, see “Client Properties” on page 39.

Table 12: PluginLink Bean Properties

Name	Description	Mandatory?
description	A detailed description of the application displayed for the end-user.	Yes
path	If provided this value will be used as the Launch link. Specifying this value overrides any automatic behavior. See screenshots at the beginning of “Displaying Links to the Application” on page 51 to see how your application will appear on the App page if this optional property is included.	No
image	The icon used to launch the application.	Yes
role	The role allowed to view the link when logged into the PureWeb server.	Yes

Application-Specific Files Overview

After completing the procedures in this chapter, you may have added files in the following locations:

- `[Installed_directory]\conf\[application name]-plugin.xml`
- `[Installed_directory]\conf\[application name]-plugin.properties`
- `[Installed_directory]\webapp\[application name]\`
- `[Installed_directory]\webapp\themes\[application name]\`
- `[Installed_directory]\webapp\WEB-INF\views\[application name]\`

If you need to uninstall the application, ensure that all of these files get removed.

Note that you must stop the PureWeb server before removing an installed application.

Chapter

9 Logs

This chapter describes the low-level logs available in the PureWeb server for troubleshooting purposes. It also describes how to configure these logs and use them for monitoring.

In addition to these logs, PureWeb developers can also use the client API's trace feature. The trace messages are displayed in the client application's Diagnostics Panel. For more information, see the Debugging chapter of the *PureWeb Developer's Guide*.

Viewing Logs

When you are logged in, you can click on the **Logs** menu option of the server's interface to access the log files.

Some logs are current (.log file extension), some are archived (.zip file extension).

To view a current log, simply click on its link to display its content.

To view an archived log, click on its link to download it locally, then uncompress the .zip file and click on the log within it to display its content in a text editor.

Logs are archived every day. You can configure the retention period; see "Log File Retention Period" on page 57.

The table below provides a brief description of each type of log.

Location	Description
pureweb.log	Contains all messages from the server about startup, shutdown, configuration details, and client activities. It also contains messages from the service application. For these, PureWeb uses the logging features available in the platforms used for service development. Anything a developer writes to the standard input/output is logged to `pureweb.log`. Messages written to this log file will indicate their severity as ERROR, WARN, INFO or DEBUG. For instructions on how to set the severity level, see “Log Severity Levels” on page 56.
access.log	Provides server access information. Messages logged to this file will also appear in the pureweb.log.
error.log	Provides a summary of any errors and warnings logged by the server. Messages logged to this file will also appear in the pureweb.log.
activity.log	Provides a summary of application activity logged by the server and cooperating applications. This includes application startup, shutdown, and events deemed to be significant by the specific application. Messages logged to this file will also appear in the pureweb.log.
catalina.log host-manager.log manager.log	These are logs defined in the default tomcat configuration. They are not directly used by PureWeb.

You can also access the log files directly in the following folder:
`[Installed_location]\Server\tomcat\logs`

Configuring Logs

The server's Configuration menu options provides the logging configuration files, where you can define the logs default retention policy, as well as the log's severity level.

Log Severity Levels

Log severity levels allow developers and server administrators to adjust how much information is provided in the `pureweb.log` file. PureWeb uses the logback logging framework to set the log severity levels (INFO, WARN, ERROR, DEBUG). For more information on this framework, including a description of each of the supported logging levels, refer to the logback API documentation at the following location:

logback.qos.ch/apidocs/ch/qos/logback/classic/Level.html

The default logging level is DEBUG. To change this level:

1. Log into the server and click on the **Configuration** link in the menu at the top to open the Configuration page.
2. Scroll down the page until you see the **Logging Configuration** section.
3. Click the `logback.xml` file link to display its content.
4. Scroll down the file until you see the element whose logging level you want to change, for example `RequestLoggingFilter`:

   ```
   <logger name="pureweb.servlet.RequestLoggingFilter">
   <appender-ref ref="access"/>
   </logger>
   ```

5. Add or edit the `level` parameter, for example:

   ```
   <logger name="pureweb.servlet.RequestLoggingFilter"
   level="INFO">
   ```

6. Click the **Save** button; you will be returned to the server's Configuration page.
7. Click the **Reload Logging** button under the section Logging Configuration for the changes to take effect.

Any configuration file that has pending configuration changes will display reload required indicating that changes have not been applied.

Log File Retention Period

The default retention policy is to keep the most recent 14 days of log files. Production installations that require different retention policies can change this configuration to suit their requirements.

1. Log into the server and click on the **Configuration** link in the menu at the top to open the Configuration page.
2. Scroll down the page until you see the **Logging Configuration** section.
3. Click the `logback.properties` file link to display its content.
4. Enter the number of retention days for each log file.
5. Click the **Save** button; you will be returned to the server's Configuration page.
6. Click on the **Reload Logging** button under the section Logging Configuration for the changes to take effect.

   Any configuration file that has pending configuration changes will display reload required indicating that changes have not been applied.

Monitoring the Application Log

Use a server monitoring tool such as Nagios (www.nagios.org) to monitor the `pureweb.log` for `glewInit()` errors using the `grep` command.

The `pureweb.log` file is located at [Installed_directory]`\Server\tomcat\logs\pureweb.log`

If the `grep` command returns a result for `glewInit()`, see example below, the monitoring tool can send an alarm stating that a GPU may be non-functional.

```
2011-05-26 07:49:06,668 DEBUG Application terminate called after
throwing an instance of 'glewException'
2011-05-26 07:49:06,669 DEBUG Application what(): glewInit
failed, this is unrecoverable
2011-05-26 07:49:06,669 DEBUG ProcessMonitor exit with status
134
```

Index

D

E

I

L

M

N

P

R

S

U

V