ICESOFT TECHNOLOGIES INC.

ICE FACES Direct-to-DOM Technology™

ICEfaces™ Enterprise Edition

Developer's Guide Supplemental

1.0 Alpha

Copyright

ICEsoft Technologies, Inc.
Suite 300, 1717 10th Street NW
Calgary, Alberta, Canada
T2M 4S2

Toll Free: 1-877-263-3822 (USA and Canada)
Telephone: 1-403-663-3322
Fax: 1-403-663-3320

For additional information, please visit the ICEsoft website: http://www.icesoft.com

Enterprise Edition 1.0 Alpha

April 2006

About this Guide

The **ICEfaces Enterprise Edition Developer's Guide Supplemental** is a guide to developing ICEfaces enterprise-scale applications, and should be used as a supplement to the **ICEfaces Developer's Guide**. By reading through this guide, you will:

- Gain a basic understanding of the additional features available in the Enterprise Edition (EE) of ICEfaces.
- Configure the Asynchronous HTTP Server and Apache Web Server for asynchronous mode application deployments.
- Configure connection management for asynchronous mode application deployments.

For more information about ICEfaces products, visit the ICEfaces page at:

http://www.icesoft.com/products/icefaces.html

In this guide...

This guide contains the following chapters organized to assist you with using features available in ICEfaces EE:

Chapter 1: Introduction to ICEfaces Enterprise Edition — Provides an overview of the features available in ICEfaces EE.

Chapter 2: Asynchronous HTTP Server — Discusses the Asynchronous HTTP Server architecture and provides instructions for configuring the Asynchronous HTTP and Apache HTTP Servers.

Chapter 3: Connection Management — Provides detailed information for connection management and configuration parameters for asynchronous communications mode.

Prerequisites

ICEfaces Enterprise Edition (EE) is an extension to the **ICEfaces Community Edition** (CE), and as such, a thorough understanding of ICEfaces CE is a prerequisite for working with the ICEfaces EE.

ICEfaces Documentation

Additional ICEfaces documentation that may be useful includes:

- **ICEfaces Getting Started Guide** — Includes information to help you configure your environment to run sample applications and a tutorial designed to help you get started as quickly as possible using ICEfaces technology.
- **ICEfaces Developer's Guide** — Provides a basic understanding of the ICEfaces architecture, key concepts, and reference information.This guide is relevant to both the Community and Enterprise Editions.
- **ICEfaces CE Release Notes** — The ICEfaces CE Release Notes provide details about new features released for ICEfaces CE.
- **ICEfaces EE Release Notes** — Read the ICEfaces EE Release Notes to learn about the new features included in this release of ICEfaces EE.
- **ICEface Online Reference (http://www.icesoft.com/support/icefaces_docs.html)** — Contains a complete listing for all ICEfaces documentation and links to ICEfaces-relevant online reference material.

ICEfaces Technical Support

For more information about ICEfaces or other ICEsoft products, visit our Technical Support page at:

http://www.icesoft.com/support/index.html

ICEfaces™ Enterprise Edition Developer's Guide Supplemental

Contents

Chapter 1 Introduction to ICEfaces Enterprise Edition

ICEfaces Enterprise Edition (ICEfaces EE) is based on the **ICEfaces Community Edition** (ICEfaces CE) augmenting that product with several key capabilities. The philosophy behind ICEfaces CE is to provide complete rich application development capabilities to the developer. ICEfaces Community and Enterprise Editions share the same core framework, component suite, and public APIs. Applications developed with ICEfaces CE and 100% compatible with ICEfaces EE. All ICEfaces CE documentation is relevant to ICEfaces EE, so you should be familiar with that documentation prior to investigating ICEfaces EE features described in this document.

The key differentiators of EE over CE relate to large-scale enterprise deployments of ICEfaces applications. In particular, the following features are described in this document:

- **Asynchronous HTTP Server Configuration and Deployment —** The Asynchronous HTTP Server is required to support large-scale deployments of asynchronous mode ICEfaces applications. Refer to *Chapter 2, Asynchronous HTTP Server,* for details describing configuration of the server and front-end Apache web server.
- **Connection Management and Heartbeating —** ICEfaces EE provides enhanced management of client/server connections. Key features like heartbeating, improved robustness of connections in enterprise deployments, and additional error handling facilities are also available. Refer to *Chapter 3, Connection Management,* for details describing heartbeat configuration and connection error handling.

Development Considerations

ICEfaces EE and CE share a common runtime environment. The following table is a summary of the JAR files included in the ICEfaces /lib directory.

JAR File	Description	Version	Source
backport-util-concurrent.jar	Backport of JSR 166 concurrent API to Java 1.4	2.0_01	http://dcl.mathcs.emory.edu/util/backport-util-concurrent/
catalina-ant.jar	Apache Tomcat Mgmt. Ant tasks	1.4	http://tomcat.apache.org/
commons-beanutils.jar	Jakarta commons-beanutil	1.6	http://jakarta.apache.org/commons/beanutils/
commons-collections.jar	Jakarta commons-collections	3.1	http://jakarta.apache.org/commons/collections/
commons-digester.jar	Jakarta commons-digester	1.5	http://jakarta.apache.org/commons/digester/
commons-discovery.jar	Jakarta commons-discovery	1.0	http://jakarta.apache.org/commons/discovery/

JAR File	Description	Version	Source
commons-el.jar	Jakarta commons-expression-language	1.0	http://jakarta.apache.org/commons/el/
commons-fileupload-1.0.jar	Jakarta commons-fileupload	1.0	http://jakarta.apache.org/commons/fileupload/
commons-logging.jar	Jakarta commons-logging	1.0.4	http://jakarta.apache.org/commons/logging/
commons-validator-1.2.0.jar	Jakarta commons-validator	1.2	http://jakarta.apache.org/commons/validator/
el-api.jar	The standardized EL specification that is standalone from JSP or JSF 1.2, but used by both frameworks.	Pre-release	http://glassfish.dev.java.net/
el-ri.jar	The standardized EL reference implementation that is standalone from JSP or JSF 1.2, but used by both frameworks.	Pre-release	http://glassfish.dev.java.net/
icefaces.jar	ICEfaces framework	1.0 Beta	ICEfaces Community Edition
icefaces-enterprise.jar	ICEfaces Enterprise Edition (only)	1.0 Alpha	ICEfaces Enterprise Edition
icefaces-facelets.jar	ICEfaces modified jsf-facelets.jar	1.11	ICEfaces Community Edition
jasper-compiler.jar	Apache Tomcat Jasper 2 compiler	5.0	http://tomcat.apache.org/
jasper-runtime.jar	Apache Tomcat Jasper 2 runtime	5.0	http://tomcat.apache.org/
jsf-api.jar	Sun JSF 1.1 RI APIs	1.1_01	http://java.sun.com/javaee/javaserverfaces/
jsf-impl.jar	Sun JSF 1.1 RI implementation	1.1_01	http://java.sun.com/javaee/javaserverfaces/
jsp-api.jar	Sun Java API for JSP	2.0	http://java.sun.com/products/jsp/
jstl.jar	Sun JSP Standard Tag Library API	1.1.0-D13	http://java.sun.com/products/jsp/jstl/
myfaces-api.jar	MyFaces JSF Runtime API	1.1.1	http://myfaces.apache.org
myfaces-impl.jar	MyFaces JSF Runtime implementation	1.1.1	http://myfaces.apache.org
servlet-api.jar	Sun Java Servlet API	2.4	http://java.sun.com/products/servlet/
tomahawk.jar	MyFaces Tomahawk Components	1.1.1	http://myfaces.apache.org
xercesimpl.jar	Apache Xerces2 Java Parser	2.6.2	http://xerces.apache.org/xerces2-j/
xml-apis.jar	Apache Xerces2 Java Parser	2.6.2	http://xerces.apache.org/xerces2-j/
z-icefaces-components.jar	ICEfaces Component Suite	1.0 Beta	ICEfaces Community Edition

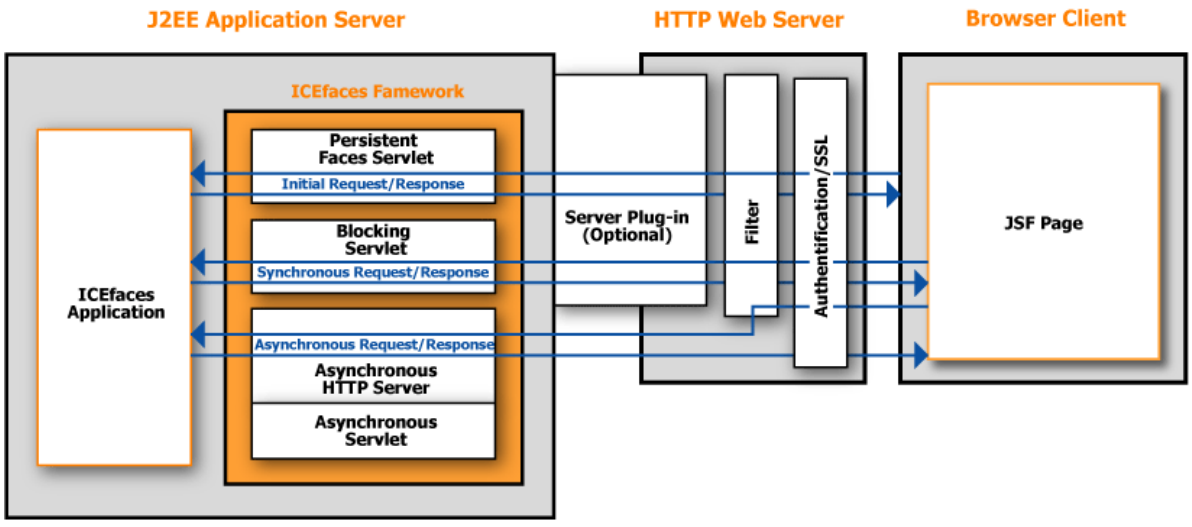
In order to include enterprise features to any ICEfaces application, it is necessary to add the enterprise JAR to the deployment. If you are working with an ICEfaces IDE integration bundle, you will need to manually add the additional JAR to your environment.

Chapter 2 Asynchronous HTTP Server

The Asynchronous HTTP Server is designed to support asynchronous mode in ICEfaces in an enterprise-scalable manner. In the standard Servlet model each outstanding asynchronous XMLHttpRequest occupies its own thread, which means that thread requirements can grow linearly with the number of clients. The long-livedness of these requests is application-specific and depends on how often server-initiated renderers occur on a per-client basis, but can have a significant impact on thread-level scalability. If server-initiated renders occur frequently, thread consumption will be somewhat mitigated, but if server-initiated renders occur infrequently, linear growth in thread consumption will result and scalability of the application will be compromised. It is strongly recommended that the ICEfaces Asynchronous HTTP Server be used in all ICEfaces application deployments where asynchronous mode is used.

Because ICEfaces also needs to maintain the synchronous communication path via a Servlet for client-initiated interactions, it is necessary to use a second port for the Asynchronous HTTP Server. Certain commercial browsers consider it a security violation to initiate HTTP requests to different ports on the same server, so it is necessary to introduce a web server to the front-end of the deployment that can direct asynchronous requests to the Asynchronous HTTP Server appropriately. The basic deployment architecture for asynchronous mode ICEfaces applications is illustrated in Figure 1.

Figure 1 Asynchronous Mode Basic Deployment Architecture



The following sections describe in detail the configuration changes required to establish a scalable, asynchronous ICEfaces application deployment with the Asynchronous HTTP Server.

An example of the configuration described in the section can be found in the ICEfaces EE under tutorials/timezone-enterprise. This example takes the Timezone 5 tutorial found in the **ICEfaces Getting Started Guide**, and converts it to an ICEfaces enterprise deployment using the Asynchronous HTTP server, and an Apache web server front end.

Configuring the Asynchronous HTTP Server

Using the Asynchronous HTTP Server with an ICEfaces application requires the following configuration changes.

1. Indicate that the Enterprise Edition (EE) asynchronous server is used by adding the following code to the web.xml file:

```
<context-param>
    <param-name>com.icesoft.faces.async.server</param-name>
    <param-value>true</param-value>
</context-param>
```

> **Note:** The Blocking Servlet must not be removed from the web.xml file as it is still responsible for handling synchronous requests.

2. The server is created and started by the Asynchronous Servlet, so the Servlet needs to be added and configured in the web.xml file of the application. To add the Asynchronous Servlet, add the following code to the web.xml file:

```
<servlet>
    <servlet-name>Asynchronous Servlet</servlet-name>
    <servlet-class>
        com.icesoft.faces.async.server.AsyncServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

   This will start the Asynchronous HTTP Server with the following defaults:

   - port used: 51315
   - non-blocking mode (Java's New I/O)
   - persistent mode (the use of persistent HTTP connections)
   - compression mode (the use of HTTP compression)
   - execute queue size: 30 (the number of threads used to handle incoming requests)

3. The Asynchronous HTTP Server needs to know about the session and context lifecycles. To provide the server with the appropriate information, it needs to listen for Servlet events. To achieve this, add the following code snippet to the web.xml file:


```
    <listener>
        <listener-class>
            com.icesoft.faces.util.event.servlet.ServletEventRepeater
        </listener-class>
    </listener>
```

4. Finally, add the following JARs to the web application build (if not already present):
   - backport-util-concurrent.jar
   - icefaces-ee.jar

Initialization Parameters

The defaults mentioned in the previous section should usually suffice, but they can be overridden, if required, by adding these as part of the Servlet container to the web.xml file:

```
<servlet>
  <servlet-name>Asynchronous Servlet</servlet-name>
  ...
  <init-param>
    <param-name>com.icesoft.faces.async.server.port</param-name>
    <param-value>51315</param-value><!-- integer -->
  </init-param>
  <init-param>
    <param-name>com.icesoft.faces.async.server.blocking</param-name>
    <param-value>false</param-value><!-- boolean -->
  </init-param>
  <init-param>
    <param-name>com.icesoft.faces.async.server.persistent</param-name>
    <param-value>true</param-value><!-- boolean -->
  </init-param>
  <init-param>
    <param-name>com.icesoft.faces.async.server.compression</param-name>
    <param-value>true</param-value><!-- boolean -->
  </init-param>
  <init-param>
    <param-name>com.icesoft.faces.async.server.executeQueueSize</param-name>
    <param-value>30</param-value><!-- integer -->
  </init-param>
</servlet>
```

Note: It is critical that the port number matches the port number in the Apache HTTP Server's configuration file, which is discussed in **Configuring the Apache HTTP Server** on page 6.

Configuring the Apache HTTP Server

This section elaborates on the ICEsoft recommended configurations for running ICEfaces applications on application servers with an Apache HTTP Server as the front-end. ICEfaces EE supplies the Asynchronous HTTP Server which is spawned inside the application server of choice. The Apache HTTP Server is therefore responsible for filtering the incoming HTTP Requests and forwarding the asynchronous requests to the Asynchronous HTTP Server and all other requests to the application server. A configuration such as this is mandatory to have full support for all mainstream Internet browsers. Refer to Figure 1 on page 3.

> **Note:** Throughout this section, we use **boldface** text for variables used in the code examples that should be replaced as required.

Routing Requests

To route all asynchronous requests to the Asynchronous HTTP Server and all other requests to the application server, add the following code to the Apache configuration file:

```
<IfModule mod_proxy.c>
    ProxyRequests Off
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
    ProxyPass /application-name/block/receive-updates
              http://host:port/application-name/block/receive-updates
    ProxyPass /application-name/
              http://host:port/application-name/
</IfModule>
```

The first ProxyPass directive defines that Request-URIs, which begin with the literal string "/**application-name**/block/receive-updates" where the **application-name** is the name of the ICEfaces application, will be proxied to the Asynchronous HTTP Server identified by the host name and port number in the URL:

```
http://host:port/application-name/block/receive-updates
```

The second ProxyPass directive defines that Request-URIs, which begin with the literal string "/**application-name**/" will be proxied to the application server identified by the host name and port number in the URL:

```
http://host:port/application-name/
```

Security Considerations

Authentication

To enforce authentication of the user when accessing an ICEfaces application, add the following code to the Apache configuration file:

```
<LocationMatch ^/application-name>
    AuthName "ICEfaces Member-Only Access"
    AuthType Basic
    AuthUserFile /var/www/secrets/.members
    require valid-user
</LocationMatch>
```

The <LocationMatch regex> container determines that every location (Request-URI), which begins with (^) the literal string "/**application-name**" where the **application-name** is the name of the ICEfaces application, is part of the ICEfaces Member-Only Access realm and, therefore, requires basic authentication for every user.

Secure Sockets Layer (SSL)

1. To enforce the usage of SSL when accessing an ICEfaces application, add the following code to the Apache configuration file:

   ```
   RewriteEngine On
   RewriteCond %{SERVER_PORT} ^80$
   RewriteCond %{REQUEST_URI} ^/application-name
   RewriteRule ^/(.*) https://%{HTTP_HOST}/$1 [R=301,L]
   ```

   The RewriteCond directives define the following conditions:

   - if the port address of the server (%{SERVER_PORT}) begins with (^) and ends with (&), thus exactly matches the literal string "80"; and
   - if the Request-URI (%{REQUEST_URI}) begins with (^) the literal string "/**application-name**", where the **application-name** is the name of the ICEfaces application.

   If both of the previous conditions are met, the RewriteRule directive defines how the Request-URI is rewritten to use HTTPS as follows:

   - $1 corresponds to the string found in the set of parentheses (that is, everything after the root (/)); and
   - rewrite the Request-URI to https://%{HTTP_HOST}/$1, where%{HTTP_HOST} is the server's host name.

2. After rewriting the Request-URI, force an external redirect (301 Moved Permanently) to the client (R=301).
3. Finally, tell the rewrite engine to end rule processing immediately (L), so that no other rules are applied to the last substituted Request-URI.

> **Note:** The mod_rewrite module is required to support this functionality. Refer to the Apache documentation on how to load modules: http://httpd.apache.org/docs/2.0/mod/mod_so.html.

Apache HTTP Server Plug-Ins

Plug-ins are modules that can be added to the Apache HTTP Server installation and can be configured to enable interaction between the server and the application server of choice. Typically, plug-ins can be used as a load balancer for the server by proxying the requests to the back-end application servers, or can be used to proxy requests for dynamic content to the back-end application server(s).

WebLogic Server 8.1 Service Pack 4

The following is a simple solution for installing and configuring WebLogic's plug-in for the Apache HTTP Server.

1. First, copy the mod_wl_20.so module supplied by WebLogic into the Apache HTTP Server's /module directory.
2. Add the following code to the Apache configuration file:

```
LoadModule weblogic_module modules/mod_wl_20.so
...
<IfModule mod_weblogic.c>
    WebLogicHost host
    WebLogicPort port
</IfModule>
```

For more information on how to install and configure WebLogic's plug-ins, refer to *BEA WebLogic Server - Using Web Server Plug-Ins with WebLogic Server*, which can be found at **http://e-docs.bea.com/wls/docs81/pdf/plugins.pdf**.

Routing Requests

In order for the plug-in to handle all the requests to the ICEfaces application with the exception of the asynchronous requests, add the following code to the Apache configuration file:

```
<LocationMatch ^/application-name(?!/block/receive-updates)>
    SetHandler weblogic-handler
</LocationMatch>
```

To explain the regular expression used in the <LocationMatch regex> container, the following is a breakdown:

^/**application-name**	if the location *begins with* (^) the literal string "/**application-name**" where **application-name** is the name of the ICEfaces application.

and

(?!/block/receive-updates)	*is not followed by* (?!) the literal string "/block/receive-updates".

This regular expression ensures that the following possible locations get handled by the plug-in:

- /application-name (initial request)
- /application-name/ (initial request)
- /application-name/xmlhttp/icefaces-d2d.js (part of initial request)
- /application-name/block/receive-send-updates?query (synchronous request)
- /application-name/block/send-updates?query (UI request)

But it prevents that the following possible location gets handled by the plug-in:

- /application-name/block/receive-updates?query (asynchronous request)

The previously mentioned code for routing requests without the use of a plug-in needs to be changed to the following:

```
<IfModule mod_proxy.c>
    ProxyRequests Off
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
    ProxyPass /application-name/block/receive-updates
              http://host:port/application-name/block/receive-updates
</IfModule>
```

Note: The second ProxyPass directive is removed, as all requests with the exception of the asynchronous requests are now handled by the plug-in.

Chapter 3 Connection Management

ICEfaces EE provides advanced connection management features not available in the CE. The following topics are presented in this chapter:

- **Asynchronous Heartbeating**
- **Connection Status Management**

Asynchronous Heartbeating

At the core of EE connection management is a configurable connection heartbeat mechanism. Heartbeating improves the long-term health of the connection by keeping it active, and closely monitors the connection status based on heartbeat responses. Heartbeating is configured in the application web.xml file using the context parameters in the code examples below.

The following code example defines the time in milliseconds between heartbeat messages. The default value is 10000 (10 seconds).

```
<context-param>
    <param-name>com.icesoft.faces.heartbeatInterval</param-name>
    <param-value>10000</param-value>
</context-param>
```

The following code example defines how long, in milliseconds, that heartbeat monitor will wait for a response prior to timing out. The default value is 1000 (1 second).

```
<context-param>
    <param-name>com.icesoft.faces.heartbeatTimeout</param-name>
    <param-value>1000</param-value>
</context-param>
```

The following code example defines the number of timeout/retries allowed before connection is considered failed. The default value is 3.

```
<context-param>
    <param-name>com.icesoft.faces.heartbeatRetries</param-name>
    <param-value>3</param-value>
</context-param>
```

Connection Status Management

Heartbeating enables an additional state for connections, so the available states are:

- idle
- waiting
- caution
- lost

The caution state occurs if heartbeats go missing, but retries are in progress. If the retries fail, the connection state will transition to lost, and if a retry succeeds the connection state will return to idle.

The outputConnectionStatus component in the ICEfaces Component Suite recognizes all four states, so applications can incorporate visual indicators for connection status. If connection lost status is determined, and no connection status component is present, ICEfaces displays a modal overlay indicating connection lost. A reload of the application is required at this point.

Note: Several additional connection management features, such as configurable redirection to an error page, are planned for a future release of ICEfaces EE.

Index

W

X