1. Architecture of Cyberelay Portal Server

1.1 High Level Overview

The architecture of Cyberelay portal server has four main components: *Aggregation Servlet*, *Request Process Chain*, *Portlet Container* which reside in *Portal Application*, and *Portlet Invocation Servlet* which resides in *Portlet Application*. (See Figure 1)

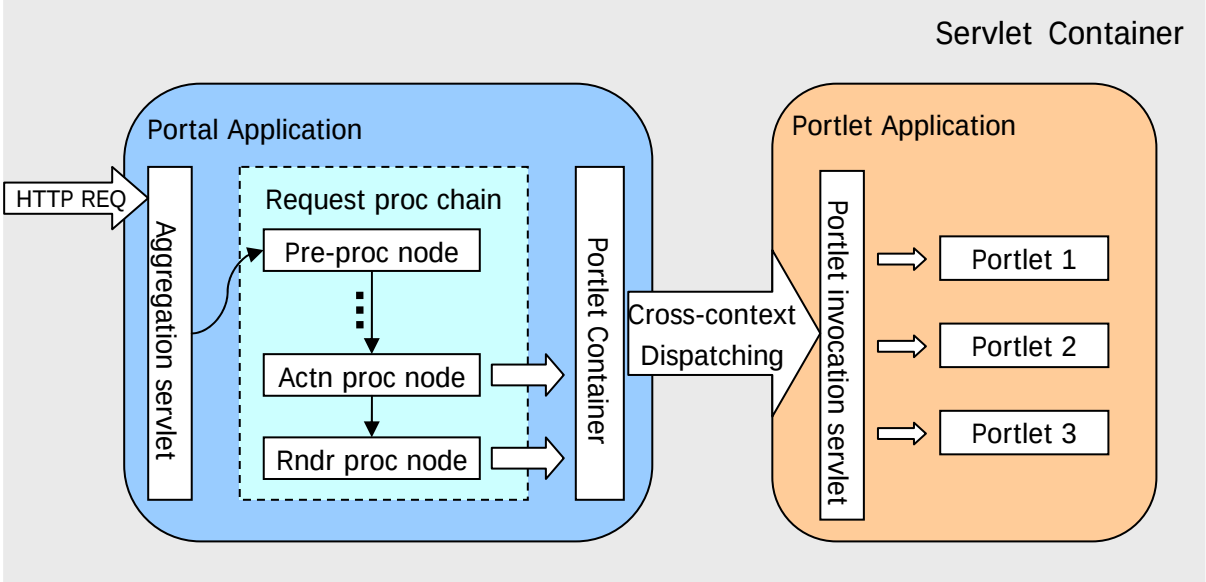

Figure1, High Level Overview

1.2 Main Components

Before we dive into the details of component, several concepts need to be introduced.

- *Portal Application.* A web application deployed on application server to receive HTTP request from client (commonly a browser) and to respond client with a portal page.
- *Portal Page.* A web page in which may have one or more portlet window embedded.
- *Portlet Window*. An area defined in a portal page whose content is generated by a designated portlet defined in portlet application.
- *Portal Request*. It refers to the HTTP request sent by client and received by Portal Application. The received Portal Request may fall into any one of the following types:
  1. *Portal page request*. It refers to a portal request that is requesting a portal page without any portlet event.
  2. *Portlet action request*. It refers to a portal request that is requesting a portal page with a portlet action event. The name of the action portlet would be specified for this type of portal request.
  3. *Portlet render request*. It refers to a portal request that is requesting a portal page with a portlet render event.
  4. *Portlet resource request*. It refers to a portal request that is requesting a portal page with a portlet resource-serving event.

5. *Logout request*. A portal request to logout Portal Application which can be treated as a special case of portal page request. The user logout of portal application differs from that of common web applications. Not only does the user session of portal applications should be invalidated, but also the user sessions of downstream portlet applications.
6. *Legacy portal request*. To support legacy jetspeed portlet API.

✧ Portlet Invocation Request. It refers to the request sent by Portal Application and received by Portlet Applications.

1.2.1 Aggregation Servlet

The primary task of *Aggregation Servlet* is to accept portal request and to pass received request to *Request Process Chain* for handling. Prior to the handling of *Request Process Chain, Aggregation Servlet* populates received portal request object with the following built-in attributes. These built-in attributes can be retrieved by calling HttpServletRequest.getAttribute() method and are used by later handling.

Attribute Names	**Description**
javax.portlet.org.cyberelay.portal.http.session	A javax.servlet.http.HTTPSession object. It is the user session object of portal application.
javax.portlet.org.cyberelay.portal.http.request	A javax.servlet.http.HttpServletRequest object. It is the received request object by Aggregation Servlet
javax.portlet.org.cyberelay.portal.http.response	A javax.servlet.http.HttpServletResponse object. It is the received response object by Aggregation Servlet
javax.portlet.org.cyberelay.portletContainer	An org.cyberelay.portletcontainer.PortletContainer object. It is the portlet container used by portal application.
javax.portlet.org.cyberelay.portal.application	An org.cyberelay.portal.PortalApplication object. It is an abstraction of portal application.
javax.portlet.org.cyberelay.portal.context	An org.cyberelay.portal.PortalContextEx object.
javax.portlet.org.cyberelay.portal.requesting.page	An org.cyberelay.portal.PageDefinition object. It refers to the target portal page under requesting.
javax.portlet.org.cyberelay.portal.requesting.url	An org.cyberelay.portal.PortalURL object. It is an abstraction of request URL of current portal request.
javax.portlet.org.cyberelay.portal.requesting.user	An org.cyberelay.portal.User object. It stands for the user of current portal request.
javax.portlet.org.cyberelay.portal.requesting.client	An org.cyberelay.portal.Client object. It stands for the request client (e.g. browser)

1.2.2 Request Process Chain

Request Process Chain is composed of a set of nodes which are chained together (See figure 2). The received portal request traverses the chain to fulfill the client request. For each node of chain, it assumes a certain task. While the received portal request exits from a certain node, its assuming task would be accomplished. The common outcome of a successful traversal would

be a generated portal page. The traversal could be terminated at any node by exceptions raised during processing. The handling of portal request is accomplished with the conclusion of portal request traversal.

There are two branch-out nodes in default request process chain. (To be added.)

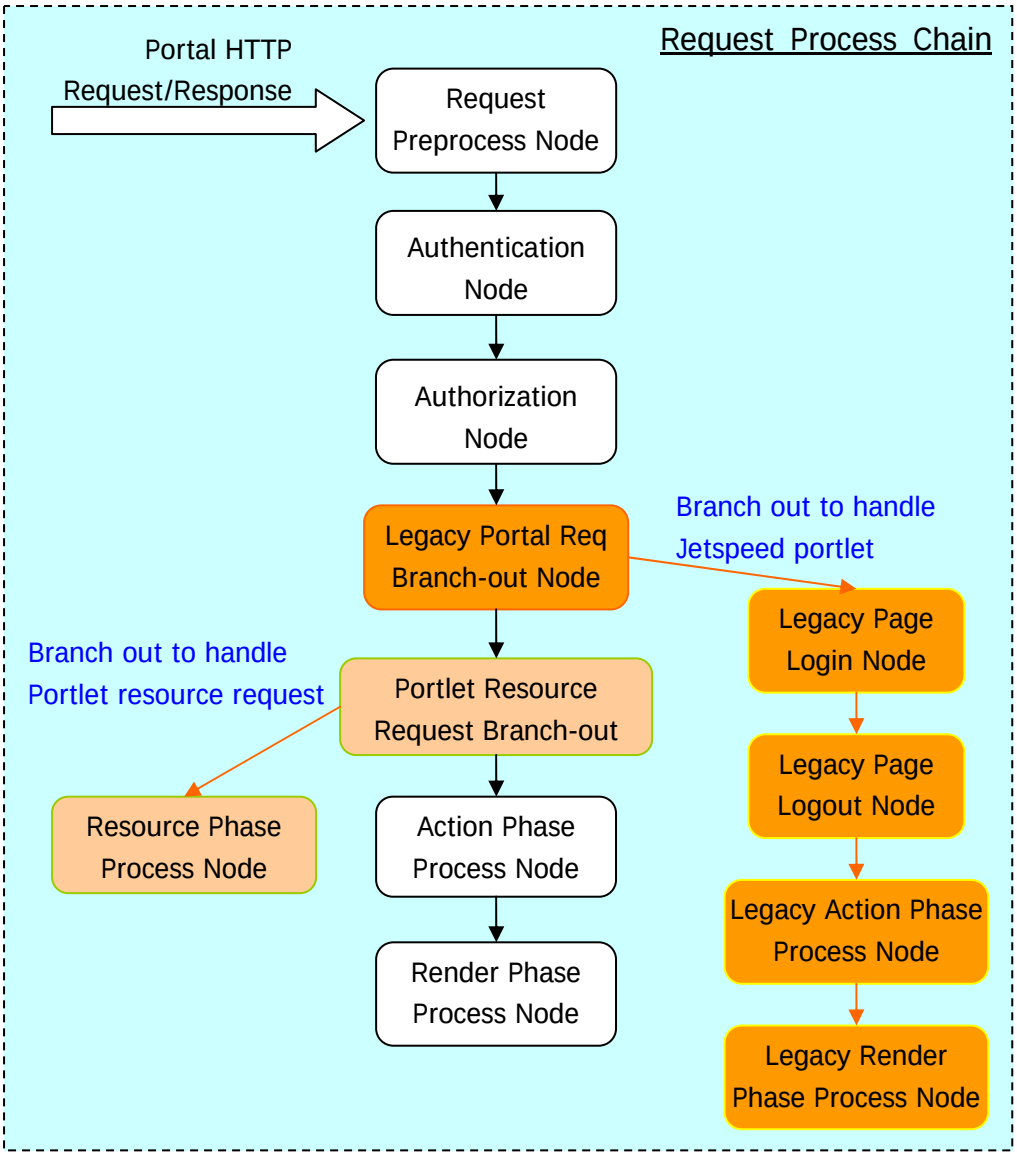

Figure2, Request Process Chain

Node Name	Responsibilities
Request Pre-process Node	✧ Logging out portal application if the incoming request is a logout request; ✧ Any other task need to be done during request pre-processing phase.
Authentication Process Node	To check if the user of the request has been authenticated. For each portal page request, the user of the request entails to be authenticated before page generation.

Authorization Process Node	To check if the access right has been granted to the authenticated user for the requesting page.
Legacy Portal Request Branch-out Node	✧ Legacy Page Login Node ✧ Legacy Page Logout Node ✧ Legacy Action Phase Process Node ✧ Legacy Render Phase Process Node (To be added.)
Portlet Resource Request Branch-out Node	✧ Resource Phase Process Node (To be added.)
Action Phase Process Node	If received portal request is a portlet action request, this node would call Portlet Container to invoke actionPerformed() method of the action portlet.
Render Phase Process Node	This node is mainly used to handle page caching. If there is no cache available or the requesting page doesn't support page caching, the node would dispatch the request/response to Page Aggregation Framework for page rendering. Otherwise, a cache page would be served. Please refer to Page Aggregation Framework section for further information.

1.2.3 Portlet Container

1.2.3.1 Portlet Container Interface

1.2.3.2 Portlet Invocation Servlet

1.2.3.3 Portlet Invocation Request Interface

1.2.3.4 Portlet Container Service

1.2.4 Page Aggregation Framework

Just like Java Swing component architecture to organize multiple UI components into one piece to realize a customized user interface, the proposed solution provided a similar approach for page aggregation. (See figure 3)

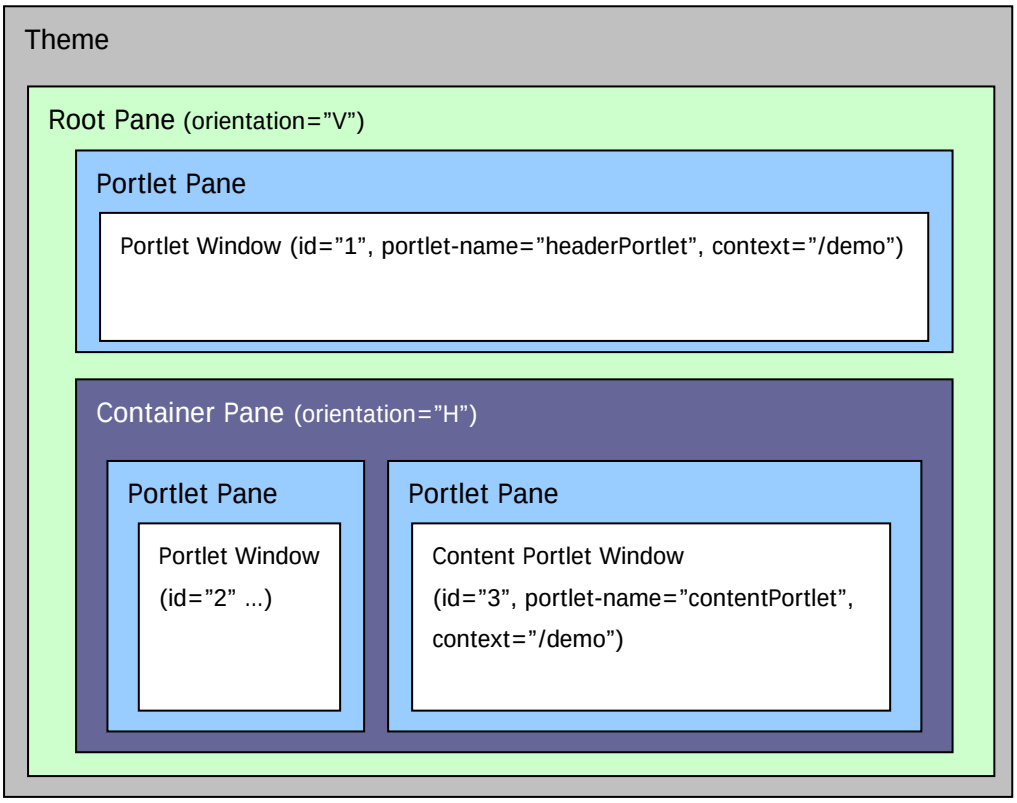

Figure3, Portal Page Aggregation

For every portal page, it is broken down into several *UIComponent*s which are commonly realized by templates such as JSP page, Velocity template etc. Two types of UIComponents are defined in Cyberelay portal solution.

- ✓ *Theme*. A UIComponent which acts as the top-level container (analogous to javax.swing.JFrame). A theme contains a root UIPane as its only child which is the main content area of portal page.
- ✓ *UIPane*. A container *UIComponent* which is analogous to javax.swing.JPanel to contain either other *UIPanes* or portlet window.
  a) *ContainerPane*. A *UIPane* that contains one or more child *UIPanes*. Every *ContainerPane* has an orientation attribute which decides if its children be lying out vertically or horizontally.
  b) *PortletPane*. A *UIPane* that contains only a portlet window. It is always *PortletPane* that is accountable for rendering portlet window controls such as Portlet minimize, maximize, help control etc.

1.2.5 Page Definition

Proposed solution leverages XML to define portal page. Below is an example.

```
<?xml version="1.0" encoding="UTF-8"?>
<portal-def>

    <!-- Template definitions -->
    <template unique-id="ThemeTemplate">
        <markup name="html" path="/template/html/Theme.jsp" />
    </template>
    <template unique-id="PortletPaneTemplate">
        <markup name="html" path="/template/html/PortletPane.jsp" />
    </template>
    <template unique-id="ContainerPaneTemplate">
        <markup name="html" path="/template/html/ContainerPane.jsp" />
    </template>

    <!-- Portlet application context definitions -->
    <portlet-context unique-id="demo.portlet.application" context-path="/demo" />

    <!-- Page definitions -->
    <page active="true" unique-id="demo.main">
        <theme template-ref=" ThemeTemplate" >
          <root-container-pane template-ref="" orientation="H">
            <container-pane template-ref=" ContainerPaneTemplate" active="true"
                ordinal="50" width="250">
                <portlet-pane template-ref=" PortletPaneTemplate" active="true"
                    ordinal="200" width="undefined">
                    <portlet-window unique-id="demo.menu"
                      portlet-name="ContentPortlet" portlet-context-ref=" demo.portlet.application" />
                </portlet-pane>
            </container-pane>
            <container-pane template-ref=" ContainerPaneTemplate" active="true"
                ordinal="100" width="undefined">
                <portlet-pane template-ref=" PortletPaneTemplate" active="true"
                    ordinal="200" width="undefined">
                    <portlet-window unique-id="demo.content"
                      portlet-name="ContentPortlet" portlet-context-ref=" demo.portlet.application" />
                </portlet-pane>
            </container-pane>
          </root-container-pane>
        </theme>
    </page>
</portal-def>
```

1.2.6 Tags for UIComponent Template

As stated above, UIComponents are commonly realized by JSP pages. The proposed solution provided the following tags to be used for UIComponent template.

Tag Name	Description
logout	To generate URL of logout portal request. It is commonly used in *Theme* template.
portletTitle	To generate portlet title. It is commonly used in *PortletPane* template.
portletEdit	To generate EDIT control. It is commonly used in *PortletPane* template
portletHelp	To generate HELP control. It is commonly used in *PortletPane* template
portletMinimize	To generate MINIMIZE control. It is commonly used in *PortletPane* template
portletRestore	To generate RESTORE control. It is commonly used in *PortletPane* template
portletMaximize	To generate MAXIMIZE control. It is commonly used in *PortletPane* template
portletRender	To invoke Portlet.render() to generate the content of portlet window. It is only used in *PortletPane* template.
if	To check if the given condition is valid. It could be used in all types of *UIComponent* template.
rootPaneRender	To generate container. It is only used in *Theme* template.

1.3 Portal Application Services

Portal Application defined the following services which would be used by its main components.

Service Name	Responsibilities
Authentication Service	✧ Authenticate the user of portal request. ✧ Extract user information from portal request. ✧ Check if user of portal request has logged in.
Authorization Service	✧ Check if the access of requesting portal page should be granted to the requesting user.
Client Info Service	✧ Extract client information (such as browser's vender, version, supporting language etc) from portal request.
Page Definition Service	✧ Identify the requesting page from portal request. ✧ Save personalized portal page. (to be supported)
Portal URL Service	✧ Parse the URL of portal request. ✧ Generate portal request URL which would be used as a link in portal pages.
Portlet Invocation Service	✧ Invoke portlet.

1.4 Portal Application Bootstrapper

1.5 Portlet Application BootStrapper