Siebel Brightware™

Implementation Guide
Concierge, Converse, and Secure Email Portal

Version 8.1.6

ORACLE®

Table of Contents

About This Guide

This manual describes how to integrate Oracle's Siebel Brightware components and your Web site. It is intended for anyone involved in integrating Concierge (web self-service) with a Web site, configuring Converse (chat), and configuring the secure portal. This includes, but is not limited to the:

- IT Administrators
- Database Administrators
- Web Administrators\Designers
- Knowledge Engineers

It is assumed that anyone using this guide is experienced with Java and Java Server Pages (JSPs).

How this Guide is Organized

Chapter 2, "Configuring Concierge (Web Self-Service)", on page 5 provides information on integrating through a Web channel via JSPs.

Chapter 3, "Configuring Converse (Chat)", on page 21 provides information on integrating a Chat channel.

Chapter 4, "Configuring the Secure Portal", on page 27 provides information on integrating the Siebel Brightware secure e-mail portal.

Chapter 5, "Configuring Brightware for Proxying", on page 35 provides information on setting up Siebel Brightware for proxying.

Related Documentation

For more information about Siebel Brightware please see these documents, included in PDF format on the Siebel Brightware CD:

- *Installation Guide*
- *Agent Guide*
- *Contact Center Console Guide*
- *Knowledge Engineer Handbook*
- *Integration Development Kit Guide*
- *Analytics Overview Guide*
- *DB Administrator Guide*

- *Report Developer Guide*

Conventions

The following typographic conventions are used in this document:

- Items that you are instructed to click or select, such as button names and hyperlinks, are bold:
  - Select **Add Response**.
  - Click the **OK** button.

- Documents, headings, and chapter titles are italicized:
  - "Refer to the *Reference Manual* for more information."

- Notes are flagged along the left margin:

This icon indicates noteworthy information.

- Cautions are flagged along the left margin:

This icon indicates critical information.

- Programming code and system messages appear in a fixed-width font:

  ```
  Set-request-condition (<condition>)
  ```

- Hyperlinks and Cross References - If viewing a document online, you can navigate through it using hyperlinks, which appear in blue text, and cross references. Although not displayed in blue, the Table of Contents and Index entries are also hyperlinks. Cross references are specific page number references. Click the page number to navigate to that page:
  - Refer to "If You Need Help", on page 2.

- The term Type usually refers to typing information on your keyboard:
  - Type the number of decimal places you want displayed.

- The term Enter typically refers to the "Enter" key on your keyboard:
  - Type the number of decimal places you want displayed and press the **Enter** key.

- When a directory path is given, the hard drive letter is omitted since it is unknown what hard drive the system is installed on. Only the default install path is supported:
  - Documents are available under edocs\Brightware\docs\.

If You Need Help

Technical Support is available to customers who have an active maintenance and support contract with Oracle. Technical Support engineers can help you install, configure, and maintain your Oracle application.

This guide contains general troubleshooting guidelines intended to empower you to resolve problems on your own. If you are still unable to identify and correct an issue, contact Technical Support for assistance.

Information to Provide

Before contacting Oracle's Siebel Technical Support, try resolving the problem yourself using the information provided in this guide. If you cannot resolve the issue on your own, be sure to gather the following information and have it handy when you contact technical support. This enables your Oracle support engineer to more quickly assess your problem and get you back up and running more quickly.

Please be prepared to provide Technical Support the following information:

Contact information

- Your name and role in your organization.
- Your company's name
- Your phone number and best times to call you
- Your e-mail address

Product and platform

- In which Oracle product did the problem occur?
- What version of the product do you have?
- What is your operating system version? RDBMS? Other platform information?

Specific details about your problem

- Did your system crash or hang?
- What system activity was taking place when the problem occurred?
- Did the system generate a screen error message? If so, please send us that message. (Type the error text or press the Print Screen button and paste the screen into your email.)
- How did the system respond to the error?
- What steps have you taken to attempt to resolve the problem?
- What other information would we need to have (supporting data files, steps we'd need to take) to replicate the problem or error?

Problem severity

- Clearly communicate the impact of the case (Severity I, II, III, IV) as well as the Priority (Urgent, High, Medium, Low, No Rush).
- Specify whether the problem occurred in a production or test environment.

Contacting Oracle's Siebel Technical Support

You can contact Technical Support online, by email, or by telephone.

Worldwide Support Center

Telephone: 800-214-0400 or 650-341-0700

Oracle's Siebel SupportWeb

https://ebusiness.siebel.com/supportweb/

Email Support

mailto: siebelsupport@oracle.com

Configuring Concierge (Web Self-Service)

About this Chapter

The primary means of integrating Siebel Brightware with a Web site is through JSP files. The JSP templates are located at
<Installdir>\edocs\Brightware\config\eservice\applications\DefaultWebApp_myserver\concierge\jsp.

In addition to these files, you can request a set of example files known as the Coldwater.com Financial sample. Contact Siebel Brightware Technical Support if you would like more information about the Coldwater.com Financial sample.

This chapter provides a reference for the content of the standard JSP templates, which provide a starting point for integration with your Web site. Reviewing these JSPs and the Coldwater.com JSPs provide the best method for understanding Siebel Brightware/Web site integration.
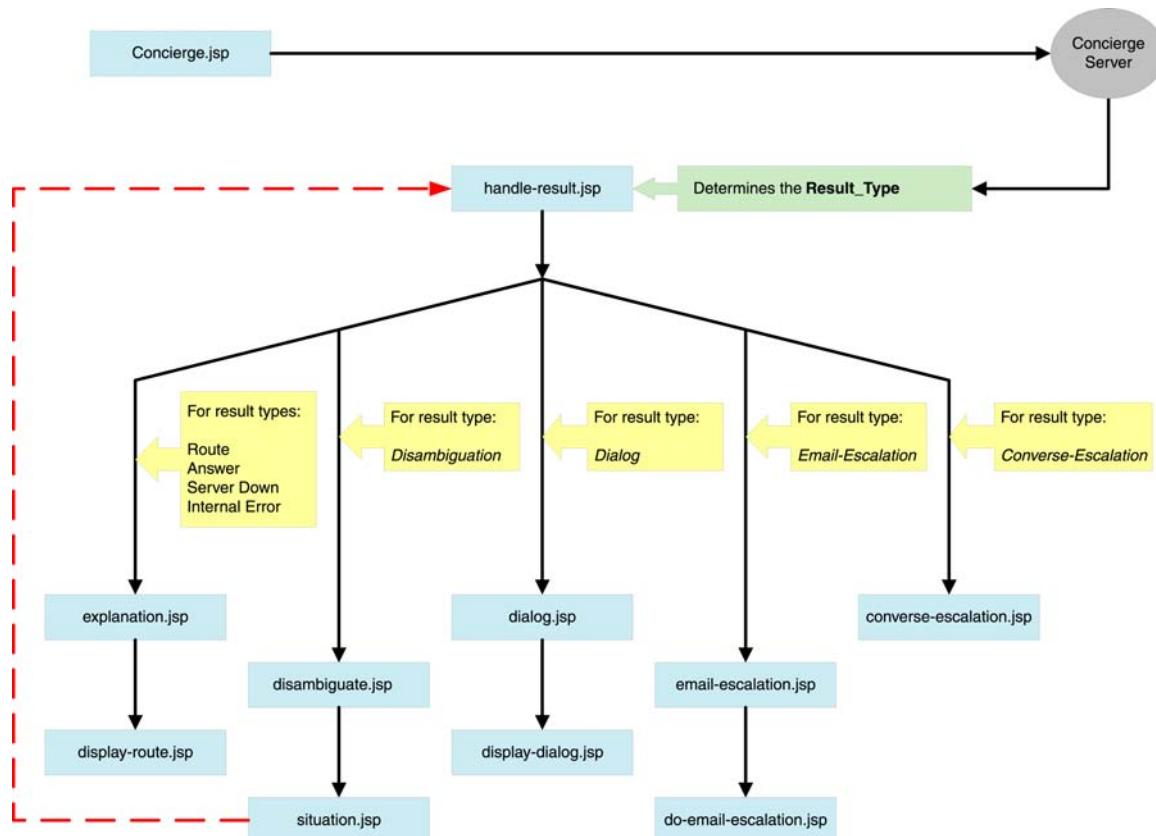
Figure 2-1. The standard JSP template information flow

Concierge.jsp

The Concierge JSP is used to start web self-service sessions. It is often triggered by an "Ask a question" button embedded on Web pages. Note the lines used for setting the RequestSource and implementing the handleQuestion method.

> **'** The setKB arguments, Initial (a business unit name) and myKB (the name of the Knowledge Base to use), must be set appropriately for the system on which Concierge is running. Create Web Rules in the Knowledge Manager within the specified Knowledge Base to apply to incoming Web messages.

Code Sample 2-1. Concierge.jsp

```jsp
<html>
<head>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>


<jsp:useBean id="concierge"
class="com.firepond.concierge.bean.ConciergeSessionBean" scope="session"/>


<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="com.firepond.results.*" %>
<%@ page import="java.util.*" %>
<%! Result result; %>
<%
// concierge.checkInProgress(false, response, "timeout.jsp");
// Store the initial question for possible use later (i.e. in email
escalation)
request.setCharacterEncoding("UTF-8");
String question = request.getParameter("question");
session.setAttribute("initial.question", question);
concierge.setRequestSource("Default");
concierge.setLanguage(ConciergeLanguage.ENGLISH_US);
concierge.setKB("Initial", "/mykb");
result = concierge.handleQuestion(request);
%>


<jsp:include page="handle-result.jsp"/>


</html>
```

handle-result.jsp

The handle-result.jsp determines which ResultType Concierge has returned. It then passes the message to the JSP which is appropriate for handling that ResultType.

Code Sample 2-2. handle-result.jsp

```
<html>
<head>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>

<jsp:useBean id="concierge"
    class="com.firepond.concierge.bean.ConciergeSessionBean"
    scope="session"/>

<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="com.firepond.results.*" %>
<%@ page import="java.util.*" %>
<%! Result result; %>
<%
concierge.checkInProgress(true, response, "timeout.jsp");
result = concierge.getLastResult();
%>

<% if ( result.getResultType() == Result.TYPE_ROUTE ) { %>
    <jsp:forward page="explanation.jsp"/>

<% } else if ( result.getResultType() == Result.TYPE_ANSWER ) { %>
    <jsp:forward page="explanation.jsp"/>

<% } else if ( result.getResultType() == Result.TYPE_SERVER_DOWN ) { %>
    <jsp:forward page="explanation.jsp"/>

<% } else if ( result.getResultType() == Result.TYPE_INTERNAL_ERROR ) { %>
    <jsp:forward page="explanation.jsp"/>

<% } else if ( result.getResultType() == Result.TYPE_DIALOG ) { %>
    <jsp:forward page="dialog.jsp?process=no"/>

<% } else if ( result.getResultType() == Result.TYPE_DISAMBIGUATION ) { %>
    <jsp:forward page="disambiguate.jsp"/>

<% } else if ( result.getResultType() == Result.TYPE_EMAIL_CHANNEL_ROUTE ) {
%>
```

```
        <jsp:forward page="email-escalation.jsp"/>

    <% } else if ( result.getResultType() == Result.TYPE_CONVERSE_CHANNEL_ROUTE )
    { %>
        <jsp:forward page="converse-escalation.jsp"/>

    <% } %>
    </html>
```

explanation.jsp

The explanation.jsp is used to resolve four different ResultTypes.

Code Sample 2-3. explanation.jsp

```
<html>
<head>
<title>Concierge Explanation Page</title>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>


<jsp:useBean id="concierge"
class="com.firepond.concierge.bean.ConciergeSessionBean" scope="session"/>
<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="com.firepond.results.*" %>
<%@ page import="java.util.*" %>


<body>
  <table border="0" width="703">
  <tr>
    <td width="34" align="center" valign="top"><img border="0"
src="../images/ask-a-question.gif" width="34" height="28" align="top"></td>
    <td width="611" align="center" colspan="3" valign="top">
      <p align="left"><img border="0" src="../images/firepond/Firepond-
Banner.gif" width="611" height="29"><br>
      <a href="../ask.htm"><img border="0" src="../images/another2.gif"
width="152" height="18"></a></p>
    </td>
    <td width="58" valign="baseline"><font face="Verdana" size="2"><a
href="javascript:parent.closeConcierge()"><img border="0"
src="../images/b_close.gif" width="58" height="14"
align="baseline"></a></font></td>
  </tr>


    <tr>
```

```
        <td width="34"></td>
        <td width="" valign="top"> <!-- 381 -->
<!-- BEGIN JSP -->
<%
concierge.checkInProgress(true, response, "timeout.jsp");
Result result = concierge.getLastResult();


// <!-- ANSWER -->
if ( result.getResultType() == Result.TYPE_ANSWER ) {
    out.println(((AnswerResult)result).getAnswer());



// <!-- SERVER DOWN -->
} else if ( result.getResultType() == Result.TYPE_SERVER_DOWN ) {
    out.println(((ServerDownResult)result).getMessage());

// <!-- ERROR -->
} else if ( result.getResultType() == Result.TYPE_INTERNAL_ERROR ) {
%>
<p>An internal error occurred while processing your request. Please try
again.</p>
<blockquote>
<%
out.println(((InternalErrorResult)result).getMessage());
%>
</blockquote>


<%
// <!-- ROUTE -->
} else {
%>
<SCRIPT LANGUAGE="JavaScript">
   parent.eRoute("<% out.print(((RoutingResult)result).getURL()); %>");
</SCRIPT>
<jsp:include page="display-route.jsp"/>
<% } %>
<!-- END JSP -->


      </td>
<!--      <td width="227" align="center" valign="top">
   <form method="POST" action="concierge.jsp">
         <p align="center"><img border="0" src="../images/another2.gif"
width="152" height="18"><br>
```

```
            <textarea rows="2" name="question" cols="25"></textarea><br>
            <input border="0" src="../images/b_submit.gif" name="I3" type="image"
   width="58" height="14"></p>
      </form>
   </td> -->
        <td width="58" valign="middle"></td>
      </tr>
      <tr>
        <td width="767" align="center" valign="middle" colspan="4">
         <p align="left"><img border="0" src="../images/firepond/powered-by-
   firepond.jpg" align="texttop" width="101" height="51">
         <font face="Verdana" size="1">Copyright &copy; 200_ edocs, Inc. All
         Rights Reserved.</font></p>
        </td>
      </tr>
   </table>


</body>
</html>
```

display-route.jsp

The display-route.jsp has two primary functions: It carries out the routing specified by Route
ResultType, and it displays additional URLs as links on the page.

Code Sample 2-4. display-route.jsp

```
<html>
<head>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>

<jsp:useBean id="concierge"
class="com.firepond.concierge.bean.ConciergeSessionBean" scope="session"/>

<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="com.firepond.results.*" %>
<%@ page import="java.util.*" %>


<%
    RoutingResult route = (RoutingResult)concierge.getLastResult();
    out.print(route.getExplanationText());
    if ( route.hasAdditionalResults() ) {
    out.print("<ul>");
```

```
    Iterator i = route.getAdditionalResults().iterator();
    while (i.hasNext()) {
        RoutingResult more = (RoutingResult)i.next();
        out.print("<li><a target=\"hostFrame\" href=\"" + more.getURL() +
"\">" +
            more.getLinkText() + "</a></li>");
    }
    out.print("</ul>");
    }

%>
<jsp:include page="feedback-form.htm"/>

</html>
```

disambiguate.jsp

The disambiguate.jsp creates a disambiguation page when the ResultType is unknown or can be one of many intents. The page is generated as set of links on the Web page, and allows the user to clarify their intent.

Code Sample 2-5. disambiguate.jsp

```
<html>
<head>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>

<jsp:useBean id="concierge"
class="com.firepond.concierge.bean.ConciergeSessionBean" scope="session"/>
<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="com.firepond.results.*" %>
<%@ page import="java.util.*" %>

<body>

<table border="0" width="700">
  <tr>
    <td width="33%" align="center" valign="top"><font face="Verdana"
size="2"><img border="0" src="../images/ask-a-question.gif" width="34"
height="28" align="top"></font></td>
    <td width="33%" align="center">
      <p align="left"><font face="Verdana" size="2"><img border="0"
src="../images/firepond/Firepond-Banner.gif" width="611" height="29"><br>
</font></td>
```

```
      <td width="34%" valign="baseline"><font face="Verdana" size="2"><img
border="0" src="../images/b_close.gif" width="58" height="14"></font></td>
  </tr>
  <tr>
    <td width="3%" align="center"></td>
    <td width="63%"><font face="Verdana" size="2"><b>There are several
      responses  with the information you requested. Please select the
option that best
      addresses your inquiry. </b></font></td>
    <td width="34%"></td>
  </tr>
  <tr>
    <td width="3%" align="center" valign="middle"></td>
    <td valign="middle"><font face="Verdana" size="2">


    <!-- BEGIN JSP -->


    <%
DisambiguationResult dr = (DisambiguationResult)concierge.getLastResult();
Iterator i = dr.possibilities().iterator();
out.print("<ul>");
while (i.hasNext()) {
    DisambiguationResult.Possibility option =
(DisambiguationResult.Possibility)i.next();
    out.print("<li><a href=\"situation.jsp?situation=" +
option.getSituationName() + "\">" + option.getDisambiguationText() +
"</a></li>");
}
out.print("</ul>");
%>


    <!-- END JSP -->


    </font></td>
    <td width="34%" valign="middle">
</td>
  </tr>
</table>

</body>
</html>
```

situation.jsp

The situation.jsp carries the situation that is selected on the disambiguate.jsp page.

Code Sample 2-6. situation.jsp

```
<html>
<head>
<title>Direct Handle Intent</title>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>


<!-- This page uses the non-EJB concierge session bean -->
<jsp:useBean id="concierge"
class="com.firepond.concierge.bean.ConciergeSessionBean" scope="session"/>
<jsp:setProperty name="concierge" property="*"/>


<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="com.firepond.results.*" %>
<%@ page import="java.util.*" %>
<%! Result result; %>
<%
concierge.checkInProgress(true, response, "timeout.jsp");
result = concierge.handleSituation(request);
%>


<jsp:include page="handle-result.jsp"/>


</html>
```

dialog.jsp

The dialog.jsp is used when the ResultType is dialog. The dialog ResultType occurs when the response from Concierge contains one or more questions. Dialog.jsp formats the questions in a Web form that can be filled out and submitted via URL parameters.

Code Sample 2-7. dialog.jsp

```
<html>
<head>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>


<jsp:useBean id="concierge"
class="com.firepond.concierge.bean.ConciergeSessionBean" scope="session"/>
```

```
<jsp:useBean id="conciergeHelper"
class="com.firepond.concierge.bean.ConciergeHelper" scope="session"/>


<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="com.firepond.results.*" %>
<%@ page import="java.util.*" %>


<%
concierge.checkInProgress(true, response, "timeout.jsp");
%>


<body>


<table border="0" width="700">
  <tr>
    <td width="33%" align="center" valign="top"><font face="Verdana"
size="2"><img border="0" src="../images/ask-a-question.gif" width="34"
height="28" align="top"></font></td>
    <td width="33%" align="center">
      <p align="left"><font face="Verdana" size="2"><img border="0"
src="../images/firepond/Firepond-Banner.gif" width="611"
height="29"></font></td>
    <td width="34%" valign="baseline"><font face="Verdana" size="2"><img
border="0" src="../images/b_close.gif" width="58" height="14"></font></td>
  </tr>
  <tr>
    <td width="3%" align="center"></td>
    <td width="63%"><font face="Verdana" size="2"><b>Concierge requires some
      additional information before it can answer your requests. Please answer
      the following questions:</b></font></td>
    <td width="34%"></td>
  </tr>
  <tr>
    <td width="3%" align="center" valign="middle"></td>
    <td valign="middle"><font face="Verdana" size="2">


    <!-- BEGIN JSP -->


    <%
    Result result = concierge.getLastResult();
    if ( ! ("no".equals(request.getParameter("process"))) ) {
        result = concierge.handleDialog(request);
    }
```

```
        if ( result.getResultType() != Result.TYPE_DIALOG ) { %>

        <jsp:include page="handle-result.jsp"/>

        <% } else { %>

        <jsp:include page="display-dialog.jsp"/>

        <% } %>

        <!-- END JSP -->

        </font></td>
        <td width="34%" valign="middle">
    </td>
      </tr>
    </table>

    </body>
    </html>
```

converse-escalation.jsp

The converse-escalation.jsp instigates an interaction with the Siebel Brightware Converse application.

Code Sample 2-8.converse-escalation.jsp

```
<html>
<head>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>

<jsp:useBean id="concierge"
class="com.firepond.concierge.bean.ConciergeSessionBean" scope="session"/>

<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="com.firepond.results.*" %>
<%@ page import="java.util.*" %>
<%
concierge.checkInProgress(true, response, "timeout.jsp");
ConverseEscalationResult result =
(ConverseEscalationResult)concierge.getLastResult();
%>
```

```
<SCRIPT LANGUAGE="JavaScript">
   parent.eRoute("<% out.print(result.getConverseURL()); %>");
</SCRIPT>


<body>


<form method="POST" action="do-email-escalation.jsp">
<table border="0" width="700">
  <tr>
    <td width="34" align="center" valign="top"><img border="0"
src="../images/ask-a-question.gif" width="34" height="28" align="top"></td>
    <td width="611" align="center" valign="top">
      <p align="left"><img border="0" src="../images/firepond/Firepond-
Banner.gif" width="611" height="29"><br>
      <a href="../ask.htm"><img border="0" src="../images/another2.gif"
width="152" height="18"></a></p>
    </td>
    <td width="58" valign="baseline"><font face="Verdana" size="2"><a
href="javascript:parent.closeConcierge()"><img border="0"
src="../images/b_close.gif" width="58" height="14"
align="baseline"></a></font></td>
  </tr>
  <tr>
    <td width="34" align="center"></td>
    <td width="611" align="center">
      <p align="left"><font size="2" face="Verdana"><b><%
out.print(result.getMessage()); %></b></font></td>
    <td width="58"></td>
  </tr>
  <tr>
    <td width="767" align="center" valign="middle" colspan="3">
      <p align="left"><img border="0" src="../images/firepond/powered-by-
firepond.jpg" align="texttop" width="101" height="51">
      <font face="Verdana" size="1">Copyright &copy; 2001 edocs, Inc. All
Rights Reserved.</font></p>
    </td>
  </tr>
</table>
</form>


</body>
</html>
```

email-escalation.jsp

The email-escalation.jsp builds a Web form that collects information for an e-mail request. When the information is submitted the user is routed to the do-email-escalation.jsp.

Code Sample 2-9. email-escalation.jsp

```
<html>

<head>
<title>Although Concierge is designed to answer up to 80</title>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>

<body>
<form method="POST" action="do-email-escalation.jsp">
<table border="0" width="700">
  <tr>
    <td width="34" align="center" valign="top"><img border="0"
src="../images/ask-a-question.gif" width="34" height="28" align="top"></td>
    <td width="611" align="center" colspan="3" valign="top">
      <p align="left"><img border="0" src="../images/firepond/Firepond-
Banner.gif" width="611" height="29"><br>
      <a href="../ask.htm"><img border="0" src="../images/another2.gif"
width="152" height="18"></a></p>
    </td>
    <td width="58" valign="baseline"><font face="Verdana" size="2"><a
href="javascript:parent.closeConcierge()"><img border="0"
src="../images/b_close.gif" width="58" height="14"
align="baseline"></a></font></td>
  </tr>
  <tr>
    <td width="34" align="center"></td>
    <td width="611" align="center" colspan="3">
      <p align="left"><font color="#000000" face="Verdana,Arial"
size="2"><b>Although Concierge is designed to answer up to 80% of inquiries
automatically, some inquiries are best handled personally by a edocs
representative. Please provide the following information so that Concierge
can escalate your request to the channel you prefer.</b></font></td>
    <td width="58"></td>
  </tr>
  <tr>
    <td width="34" align="center" rowspan="4"></td>
    <td width="69" align="left" valign="top">
      <font color="#000000" face="Verdana,Arial" size="2">Name:</font></td>

    <td width="234" align="center" valign="top">
```

```
    <!-- NAME -->
    <input name="name" size="35" tabIndex="1">
    </td>


    <td width="302" align="center" rowspan="4" valign="top">
      <p align="left"><font color="#000000" face="Verdana,Arial"
size="2">Would
                    you like to add or modify your inquiry?</font>
    <!-- MESSAGE -->
    <textarea cols="40" name="message" rows="3" style="FONT-FAMILY: Arial;
FONT-SIZE: 10pt" tabIndex="4">
    <%
    String message = (String)session.getAttribute("initial.question");
    if ( message != null )
    out.print(message);
    %>
    </textarea>


    </td>
    <td width="58" valign="top" rowspan="4"><input border="0"
src="../images/b_submit.gif" name="I2" width="58" height="14"
type="image"></td>
  </tr>
  <tr>
    <td width="303" align="left" colspan="2" valign="top">
      <font color="#000000" face="Verdana,Arial" size="2">Please
                    select how you would like to be contacted:</font></td>
  </tr>
  <tr>
    <td width="69" align="left" valign="top">
      <font color="#000000" face="Verdana,Arial" size="2">Email:</font></td>
    <td width="234" align="center" valign="top">


    <!-- EMAIL -->
    <input name="email" size="35" tabIndex="2">
    </td>
  </tr>
  <tr>
    <td width="69" align="left" valign="top">
      <!-- <font color="#000000" face="Verdana,Arial" size="2">or
    Phone:</font>--></td>
    <td width="234" align="center" valign="top">
    <!-- PHONE -->
<!--    <input maxLength="35" name="phone" size="35" tabIndex="3"> -->
```

```
        </td>
      </tr>
      <tr>
        <td width="767" align="center" valign="middle" colspan="5">
          <p align="left"><img border="0" src="../images/firepond/powered-by-
firepond.jpg" align="texttop" width="101" height="51">
          <font face="Verdana" size="1">Copyright &copy; 2001 edocs, Inc. All
Rights Reserved.</font></p>
        </td>
      </tr>
    </table>
    </form>


    </body>
    </html>
```

do-email-escalation.jsp

The do-email-escalation.jsp page uses the information gathered by email-escalation.jsp and passes it on to the appropriate location.

Code Sample 2-10. do-email-escalation.jsp

```
<html>
<head>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>


<jsp:useBean id="concierge"
class="com.firepond.concierge.bean.ConciergeSessionBean" scope="session"/>


<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="java.util.*" %>
<%
String name = request.getParameter("name");
// String phone = request.getParameter("phone");
String body = request.getParameter("message");
String email = request.getParameter("email");
Map data = new HashMap();
data.put("name",  name);
// data.put("phone", phone);
data.put("body",  body);
data.put("email", email);
concierge.setRequestSource("Do Email Escalation");
```

```
concierge.handle(data);
%>


<jsp:include page="handle-result.jsp"/>
</html>
```

do-feedback.jsp

The do-feedback.jsp collects feedback information once the other JSPs have completed their respective tasks. The information can be recorded to a database and used for analysis and reporting later.

Code Sample 2-11. do-feedback.jsp

```
<html>
<head>
<%@ page contentType="text/html; charset=utf-8" session="true" %>
</head>


<jsp:useBean id="concierge"
class="com.firepond.concierge.bean.ConciergeSessionBean" scope="session"/>


<%@ page import="com.firepond.concierge.bean.*" %>
<%@ page import="java.util.*" %>
<%
concierge.checkInProgress(true, response, "timeout.jsp");
String feedback = request.getParameter("feedback");
Map data = new HashMap();
data.put("feedback", feedback);
data.put("acknowledgement", "Thank you for using edocs Concierge. Your
feedback has been recorded.");
concierge.setRequestSource("Do Feedback");
concierge.handle(data);
%>


<jsp:include page="handle-result.jsp"/>
</html>
```

Configuring Converse (Chat)

About this Chapter

The interaction channel provides the eCustomer with the capability of interacting with your Contact Center agents via a live channel on the web. These live requests are distributed to your agents though the same queuing mechanism and use the same Agent Desktop as the e-mail channels, while the eCustomer is provided with a standard web form to ask questions and see their responses with no special download for the web client.

Routing rules can be created in the Contact Center Console to route Chat Requests to high priority queues so agents can respond quickly to eCustomers waiting at the web site, and reports can be specialized to focus on live interactions. The eCustomer also has the ability to opt-out of chat and have their responses or transcripts sent to them at their specified e-mail address.

The Chat Channel also provides the capability to identify alternative URLs to route the eCustomer during holidays or non-working hours. Requests made during these times can be routed to the Contact Center and the replies can be sent to the eCustomer though the normal e-mail channel.

Siebel Brightware provides you with a sample implementation of chat that can be used to get your chat channel up and running quickly. We also provide you with all of the HTML files so that you can further customize the sample to integrate seamlessly with your existing web site. The HTML files can be found at:
<installdir>/config/eservice/applications/DefaultWebApp_myserver/converse/.

The contents are described further in this chapter.

The Web Site Visitor Interface

The interface may be composed of several HTML fragments at any given time. The contents of the screen may change depending upon actions taken by site visitors or the program.

Users working with HTML fragments are expected to know how to write HTML code with Java scripting. In addition, they should be familiar with XSL stylesheet files and how to edit them to reformat text appearing in the chat window.
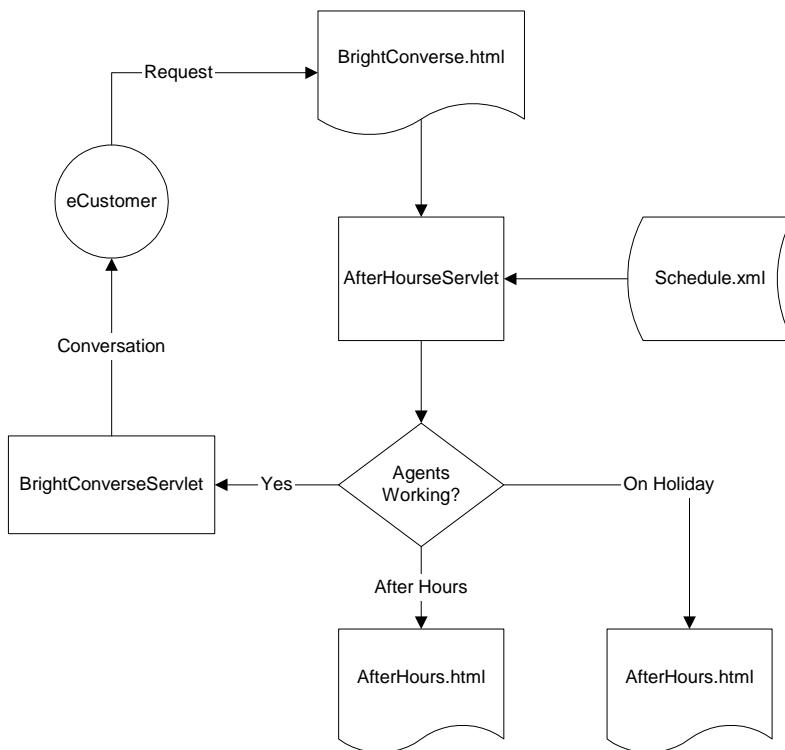
Figure 3-1. HTML Fragment Interaction

Table 3-1. Chat Interface – file that needs to be modified to point to the server (required)

File that needs to be modified to point to the server (required). Use only one.	Purpose
BrightConverse.html (default)	Entry to Converse; establishes the title and look of the screen; defines frames which will hold other files; opens in a popup window; the default configuration; see also:
BrightConversen.html (version)	Version of BrightConverse.html; opens chat in the current browser window (instead of a popup window).
BrightConversep.html (version)	Version of BrightConverse.html; includes a Push URL frame with chat in a popup window.
BrightConverser.html (version)	Version of BrightConverse.html; opens a vertical popup window where e-mail and other information are gathered (such as for registration) and the initial inquiry can be edited before opening chat in the same popup using registration.html.
BrightConversev.html (version)	Vertical version of BrightConverse.html; establishes the title and look of the screen; defines frames which will hold other files; opens in a popup window.

Table 3-2. Chat Interface—Files or Fragments with macros that you may customize (optional)

Files or Fragments with macros or JavaScript that you may customize (optional)	Purpose
browserpopupwindow.html	JavaScript code that opens a pop-up browser window for each Pushed URL in a Converse response.
close.html	Thanks the customers, and closes the chat window.
conversationstate.html	Presents the updated conversation; contains copyright information.
converseh.html (version) **conversev.html (version)**	Establishes the title and look of the conversation part of the screen; defines frames that hold other files; versions h and v display a horizontal or vertical interface respectively; set in hidden input fields in BrightConverse.html (or in n, p, r. or v versions of BrightConverse.html).
conversepopupstart.html	Asks the site visitor to wait for the exchange to begin; is only used for popups.
conversepushpopupframes.html	Used only in the Push Frame version of Converse; appears after visitors make their initial request; calls the servlet to determine elapsed time and places the result in the periodic frame, and loads all of the other frames.
email_close_gottranscript.html **email_close_wanttranscript.html**	Defines a form for e-mail address entry; notifies the visitor of the currently recorded e-mail address; captures any new e-mail address entered; gottranscript used when chat session is set to send a transcript; wanttranscript is used before gottranscript is used.
Followonreset.html	Reloads the chat screen using a script.
internalerror.html	Calls an expression that pops up an error message box.
messageaccepted.html	Returned by the initial submission after visitors submit their requests and before the session is created by the server and before the chat session appears.
periodicframe.html	Appears at the bottom of the chat screen once visitors submit their requests. It notifies visitors of the elapsed time since the submission of their request.
popupwindow.html	JavaScript code that opens a pop-up browser window for the chat session.
pushwillcomehere.html	Used only in the Push Frame version of Converse; defines message which appears before the Push URL appears.
registration.html	Optional registration form for site visitors to fill out; use only single quotes in this file (see also BrightConverser.html).
replaceframeset.html	Runs a script; calls an expression that initiates full replacement of Converse's popup contents.
replacebigwindow.html	JavaScript code that replaces the pushed frames' content.
replacetoclose.html	Runs a script; calls an expression; replaces Converse's popup contents with Goodbye screen.

Table 3-2. Chat Interface—Files or Fragments with macros that you may customize (optional)

Files or Fragments with macros or JavaScript that you may customize (optional) (continued)	Purpose
requestentryh.html (version) **requestentryv.html (default)**	Establishes frame where follow-on request text is entered; identifies the SUBMIT button; h version is horizontal version of requestentry.html; v version is vertical version; set the default in BrightConverse.html.
tickerdisplay.html	Tracks time since visitor entered request.
tomainframe.html	Writes new conversation state into the Conversation frame; initiates new polling sequence.

Table 3-3. Chat Interface — Files or Fragments that require no translation by the chat server

Files or Fragments that require no translation by the chat server and that you may customize (optional)	Purpose
conversationwillcomehere.html	Tells the site visitor that the response will appear here.
goingviaemail.html	Appears after visitors request an e-mail response; notifies visitors that chat will send their responses via e-mail.
header.html	Identifies the graphic used for the header.
pleasewaitforresponse.html	Used only in the Push Frame version of Converse; appears after visitors submit their initial request, and asks them to wait for their responses.

Table 3-4. Files read, cached and served up, but require no translation by Servlet

Files read, cached and served up by Servlet, but have no macro substitution	Purpose
mustenterfollowon.html	Alerts the visitor to enter a follow-on message instead of submitting a blank request.
mustentermessage.html	Alerts the visitor to enter a message instead of submitting a blank request.

Table 3-5. Translation Layers/Stylesheet

Stylesheet file that acts as a translation layer for text in the chat frame.	Purpose
Converse.xsl	Establishes styles for text to be used in the chat frame.

Table 3-6. Development tool

Development tool	Purpose
BrightConverseClear.html	Clears template HTML files from the caches, causes XML parser to reload Converse.xsl file used to format chat output (conversaton.html) in the chat frame (not e-mail) without you having to restart the server.

BrightConverser.html

BrightConverser.html defines the initial Web visitor interface that passes off content to the AfterHoursRedirect Servlet. It uses hidden input fields to define the size of the popup window if the popup variable is set to true. BrightConverser.html is the file most commonly used as the default initial page for chat installations.

AfterHoursRedirect Servlet

The AfterHoursRedirect Servlet uses the schedule.xml file to determine whether or not the web customer is asking a question during business hours, on a holiday, or when the server is down. If the current time is in a declared holiday date or date range, the URL for this holiday is displayed. If the time is not in the time range element corresponding to the current day of the week, the URL for this day is selected or the default URL for the weekly parameter is used. If the server is online, the URL for uptime redirection is selected, otherwise the URL for downtime redirection is selected.

Configuration of the schedule.xml file:

Servlet configuration is stored in XML file. This file will be maintained by the Contact Center Console. The first two lines of this file

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE schedule SYSTEM "schedule.dtd">
```

describe this file as an XML1.0 document with datatype definition described in file schedule.dtd. The root element in XML structure is <schedule> element. It consists of the following elements:

- **<uptime>** – Element contains attribute 'html-ref', which value is the URL for redirecting when Converse server is online.
- **<downtime>** – Element contains attribute 'html-ref', which value is the URL for redirecting when chat server is offline.
- **<weekly>** – Element contains attribute 'html-ref', which value is the URL for redirecting when server is online, but it is not working time for live ages. Also, <weekly> element contains 7 elements corresponding to days of week: <sun>, <mon>, <tue>, <wed>, <thu>, <fri>, <sat>. Each has an optional 'html-ref' attribute, which value is the URL for redirecting "after-hours" for this day of week (sunday or saturday for example). Every element contains zero, one or more <time-range> elements, which describe working time ranges for this day.
- **<time-range>** – Element contains two <time> elements, the first is the low bound for this range and the second is the high bound for this range.
- **<time>** – Element has required attribute 'hour', optional attribute 'min.' (default value is "0"), and optional attribute 'sec.' (default value is "0").
- **<holidays>** – Elements contains attribute 'html-ref', the default URL for redirecting on holidays. This element contains zero, one, or more <holiday> elements.
- **<holiday>** – Element contains attribute 'html-ref', the URL for redirecting for holidays. This element contains either one <date> element or two <date> elements. In the first case given, <date> element describes one day per holiday. In the second case given, <date> elements describe date range for a given holiday.
- **<date>** – Element has required 'day' attribute, required 'month' attribute, and optional 'year' attribute (in the second case this holiday is year-periodic).

The Schedule.xml file is located at: <Installdir>/config_/eservice/applications/ DefaultWebApp_myserver/converse

Configuring the Secure Portal

The Siebel Brightware Portal application is constructed using Struts and tags. This chapter lays out the relationship between the portal components and describes the purpose of each component.

Secure E-mail Portal Architecture

The secure e-mail portal primarily provides a secure place for eCustomers to view responses to requests they have submitted. The responses may be secure for several reasons, such as containing sensitive or confidential information. The portal can also be used by customers for submitting new requests or responding to existing requests. eCustomers can also view and search their entire history of request submissions.

The portal is implemented as multiple abstraction layers between a thin-client Graphical User Interface (GUI) front-end and the edocs domain.

- **Thin-client GUI Layer** – Java Server Pages (JSP files) specifying the layout and look-and-feel of the portal views.
- **JavaScript Layer** – This layer performs basic data validations and passes requests on to the servlet layer.
- **Servlet and Presentation Layer** – Handles portal requests from JavaScript or Java Server Pages, and makes calls to the Desktop and Application layers as needed.
- **Desktop Layer** – Contains commonly-used classes shared across all desktops.
- **Application Layer** – A session bean layer sitting on top of the domain model that provides a more desktop-centric function.
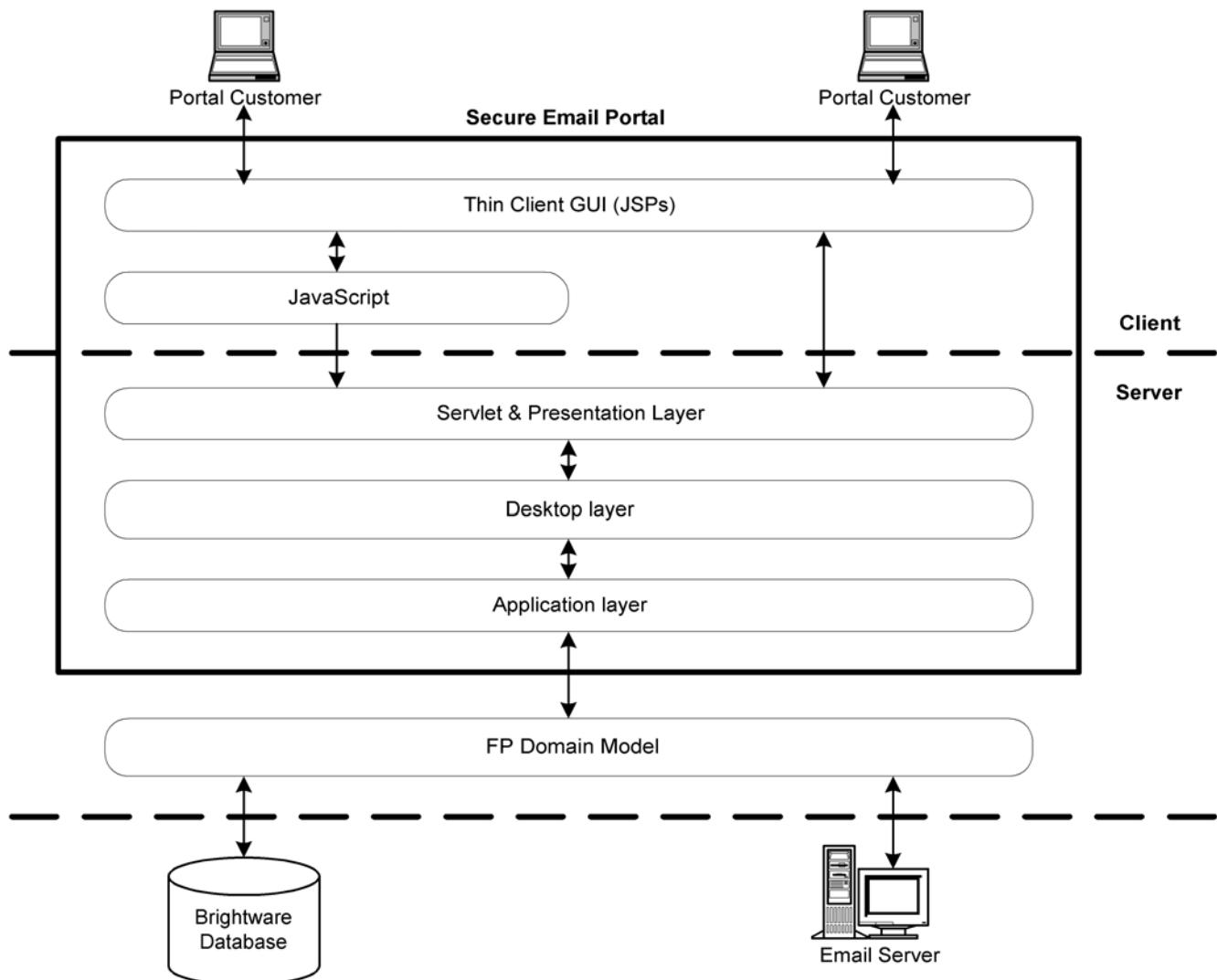
Figure 4-1. The Portal Components

Integration into an Existing Web Site

The look-and-feel of the Portal screens may be customized for seamless integration into an existing web site. This customization is accomplished by editing the Java Server Pages (JSP files) that specify the look-and-feel of the Portal. These files can be found in the portal.war file located in the /config/eservice/applications subdirectory of your edocs installation. You can extract them using an archive utility, edit them to modify the look of the Portal views, and then update the portal.war file with your modifications.

Here is a description of each JSP file:

- **logon.jsp** – The logon screen.
- **newuser.jsp** – The new user registration screen.
- **newusersuccess.jsp** – A confirmation screen that notifies a new user that registration was successful.
- **history.jsp** – The My History view that displays the history of all messages submitted by the customer.
- **msgpreview.jsp** – The message preview screen.
- **newrequest.jsp** – The New Request view.
- **requestconfirmation.jsp** – A confirmation screen notifying the customer that the request was submitted successfully.
- **requestfailure.jsp** – A confirmation screen notifying the customer that an error occurred and the request was not submitted successfully.
- **changepassword.jsp** – The My Account view.
- **passwordsuccess.jsp** – A confirmation screen notifying the customer that the password was changed successfully.
- **passwordfailure.jsp** – A confirmation screen notifying the customer that something went wrong and the password was not successfully changed.

The JSP files are mostly standard HTML, but do make use of two proprietary edocs tags which are described below.

Interaction Example - New Request

The following sequence diagram for the New Request action illustrates the various interactions that take place between the architectural layers of the Portal application.
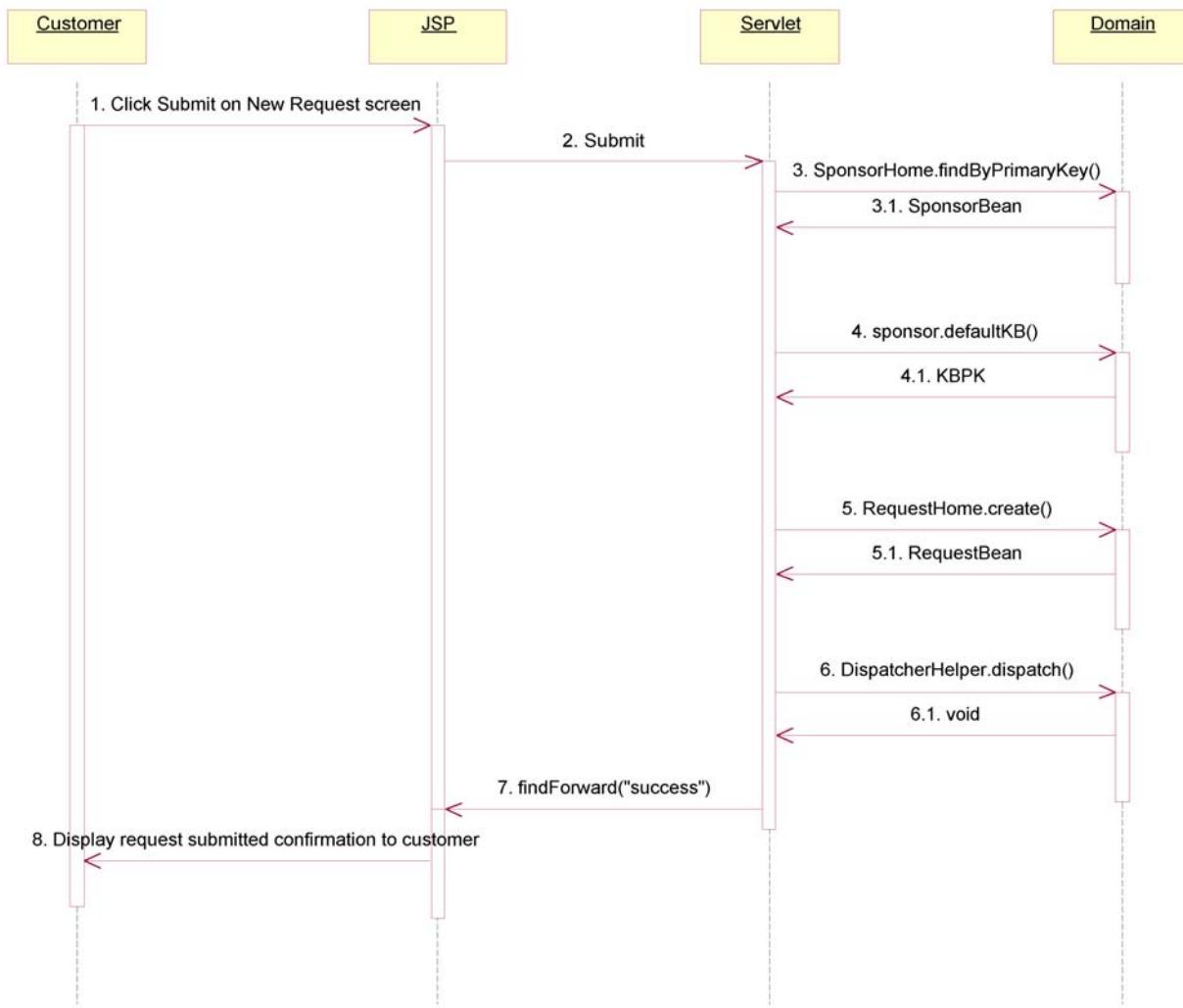
Figure 4-2. New Request Sequence

1. The customer clicks on the **Submit** button in the New Request view of the Portal to submit a new request.
2. The HREF property of the Submit link in newrequest.jsp calls submit(), which hands off the request to the process() method of the NewRequestAction servlet.
3. The NewRequestAction servlet calls the SponsorHome.findByPrimaryKey() domain method, which returns a SponsorBean.
4. The NewRequestAction servlet calls the defaultKB() method of the SponsorBean, which returns a KBPK.

5. The NewRequestAction servlet calls the RequestHome.create() domain method to create a new request. The method returns a RequestBean.

6. The NewRequestAction servlet calls the DispatcherHelper.dispatch() method.

7. The servlet calls findForward("success") to return the request confirmation JSP.

8. The customer receives a confirmation screen, indicating the request was submitted successfully.

email:msgpreview Tag

The email:msgpreview tag displays an e-mail message. For example:

Figure 4-3. email:msgpreview tag

Table 4-1. email:msgpreview parameters

Parameter	Type	Default Value	Description
msgID	integer	0	The unique ID of the message to display.
listName	string	none	The parameter name under which the dataset containing the data to be displayed is stored.
redirectTo	string	none	The URL/URI to which the Back button will return to.
borderColor	string	"#C6AD94"	The color to use for the borders of the table.
bgColor	string	"#F4F0EC"	The background color.
headerClass	string	"Prompt"	The style name (defined in a cascading style sheet) to use in displaying the title text (TO:, BCC:, etc).

email:datagrid **Tag**

The email:datagrid tag displays a grid of data. For example:

Figure 4-4. email:datagrid tag

Table 4-2. email:datagrid parameters

Parameter	Type	Default Value	Description
listName	string	none	The parameter name under which the dataset containing the data to be displayed is stored.
border	integer	1	The table border width.
borderColor	string	"#D7C5B4"	The color to use for the borders of the table.
bgColor	string	"#C6AD94"	The background color.
headerClass	string	"menu"	The style name (defined in a cascading style sheet) to use in displaying the title text (TO:, BCC:, etc).

Associating the Secure Portal with another Business Unit

The Secure Email Portal is associated with the Default Business Unit; all new requests submitted via the Portal automatically are directed there.

To associate the Portal with a different Business Unit:

1. Shut down the Siebel Brightware Server.
2. Locate the portal.war file in the /edocs/Brightware/config/eservice/applications directory of your server installation.
3. Use an archive utility (such as jar.exe or winzip) to extract the contents of the portal.war file to a temporary directory.
4. Edit the /web-inf/web.xml file extracted to the temporary directory: locate the following lines:

```
<init-param>
        <param-name>Business Unit</param-name>
        <param-value>Initial</param-value>
</init-param>
```

5. Modify the Business Unit name from Initial to the name of the Business Unit you want to associate with the portal.
6. Update the portal.war file with the newly-modified web.xml file.
7. Restart the Siebel Brightware Server.

Multiple Portal Instances

Follow this procedure to establish two different Portal instances, for example, one Portal associated with the Initial Business Unit and one associated with the Sales Business Unit.

To create multiple Portal instances:

1. Make a copy of the portal.war file and name the new copy sales.war.
2. Edit the web.xml in the sales.war, and change the business unit it is associated with from Initial to Sales.
3. Register the new sales.war file so that it will be loaded by the Siebel Brightware Server. Edit the config.xml file located in the /config/eservice subdirectory of your edocs installation: locate the following lines:

   ```
   <Application Deployed="true" Name="portal"
   Path=".\config\eservice\applications">
         <WebAppComponent Name="portal" Targets="myserver"
     URI="portal.war"/>
   </Application>
   ```

4. Copy these lines to create a similar entry for the new sales.war file. For example:

   ```
   <Application Deployed="true" Name="sales"
   Path=".\config\eservice\applications">
         <WebAppComponent Name="sales" Targets="myserver"
     URI="sales.war"/>
   </Application>
   ```

5. Save your changes and restart the Siebel Brightware Server.

Configuring Siebel Brightware for Proxying

Proxying is the process of redirecting an HTTP request from a standard Web server to the Siebel Brightware server. Siebel Brightware's Web Channel, Chat Channel, and the Secure Email Portal components can be proxied. The Contact Center Desktops are not proxyable.

Proxying by extension redirects requests on the main server to the application server based on the filename extension of the request. For example, all JSP or DO requests can be redirected to the application server, while all static content (such as plain HTML pages and images) is handled by the web server. Proxying by path maps all requests below a particular path on the web server to the application server. For example, all requests to http://web/edocs/ can be redirected to http://edocs:7001/.

Secure Email Portal and the Web channel are proxied by file extension. The Chat Channel is proxied by path.

Setting Up Proxying

- Internet Information Server (IIS) 5.0 – Instructions for proxying Microsoft IIS 5.0 are derived from the document at:
  - http://e-docs.bea.com/wls/docs61/adminguide/isapi.html
- Apache – The same procedure applies to the Apache Web Server. BEA's instructions for configuring the Apache plug-in can be found at:
  - http://e-docs.bea.com/wls/docs61/adminguide/apache.html
- Netscape Enterprise Server – The same procedure applies to the Netscape Enterprise Server. BEA's instructions for configuring the Netscape plug-in are at:
  - http://e-docs.bea.com/wls/docs61/adminguide/nsapi.html

Two Extensions

This version of Siebel Brightware has two extensions to be proxied: JSP and DO.

Since the proxying will only proxy the dynamic content (the JSP and DO URLs), the static content must be copied to the primary web server. For the purpose of discussion we will assume that the web server root directory is at:

```
C:\InetPub\wwwroot
```

and the edocs installation directory is:

```
C:\edocs\Brightware
```

Web Channel

For the Web channel, simply copy the following directory (and all its contents):

```
C:\edocs\Brightware\config\eservice\applications\
DefaultWebApp_myserver\concierge\
```

to:

```
C:\InetPub\wwwroot\concierge\.
```

Secure Email Portal

Portal is slightly more involved, as the static content is contained in an archive file. That archive file is:

```
C:\edocs\Brightware\config\eservice\applications\
portal.war.
```

1. The contents of portal.war must be extracted to:

    ```
    C:\InetPub\wwwroot\portal.
    ```

    This can be accomplished using the jar.exe command, which is located at:

    ```
    C:\edocs\Brightware\jdk131\bin\jar.exe.
    ```

2. Copy portal.war to:

    ```
    C:\inetput\wwwroot\portal\
    ```

3. Run the following command:

    ```
    jar xf portal.war
    ```

Chat Channel

The Concierge and Portal components utilize proxying by extension for all JSP and DO requests, while the Chat component utilizes proxying by path.

IIS Proxying

The Web Channel ISAPI is distributed as the dynamic-link library iisproxy.dll for NT. This file and the iisproxy.ini are located in:

```
C:\edocs\Brightware\wlserver6.1\bin
```

If you're going to reroute by file extension and your system already uses JSP's you'll need to use an alternate extension, or put your existing JSP's on the Siebel Brightware Server.

, Ordinarily the file extension would be JSP. However this presents a problem when JSP files are already in use on the Web site. To avoid sending all JSPs to the Siebel Brightware Server, this example will use ESP instead. This requires that the file extensions for all Web Channel JSP files be changed to ESP. In addition all references to these files in the Knowledge Bases and within the JSPs themselves must be altered to ESP. If you intend to use ESP in the IIS you must make these modifications or the Web Channel will not work. If you are setting up the Web Channel for the first time, it is advised that you use the JSP extension.
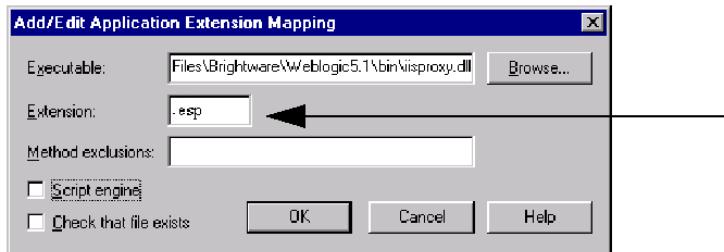
REMEMBER: Use .jsp by
default. Only use .esp if
routing around existing JSPs

Figure 5-1. Setting the file extension type

Proxying by File Extension

The ISAPI module can only proxy files that correspond to the types registered for proxying in the Internet Service Manager. If you have registered type JSP or ESP for proxying, all requests with the designated extension will be proxied to Oracle's Siebel Brightware Server. Please refer to the Microsoft IIS documentation for details.

Index

B
BrightConverseClear.html, development tool 25

C
conversationwillcomehere.htlm 24
converse.xsl 24
conversepushpopupframes.html 23

D
development tool
    BrightConverseClear.html 25
Documentation
    updates 2

E
email_close_gottranscript.html 23
email_close_wanttranscript.html 23

F
File
    Extension Proxying 37
Followonreset.html 23

G
goingviaemail.html 24

H
header.html 24
HTML files
    BrightConverseClear. 25
    Conversationwillcomehere 24
    Conversepushpopupframes.html 23
    Email_close_gottranscript 23
    Email_close_wanttranscript 23
    followonreset 23
    Goingviaemail 24
    Header 24
    Internalerror 23

Messageaccepted 23
Mustenterfollowon 24
Mustentermessage 24
Periodicframe 23
Pleasewaitforresponse 24
Popupwindow 23
Pushwillcomehere 23
Registration 23
Replacepushwindow 23
Replacetoclose 23
Requestentryh 24
Requestentryv 24
Tickerdisplay 24
Tomainframe 24

I
IIS
    Proxying 36
internalerror.html 23

M
messageaccepted.html 23
mustenterfollowon.html 24
mustentermessage.html 24

O
Obtaining Documentation Updates 2

P
periodicframe.html 23
pleasewaitforresponse.html 24
popupwindow.html 23
Proxying
    by File Extension 37
pushwillcomehere.html 23

R
registration.html 23