

ActuateOne™

One Design
One Server
One User Experience

Information Console Developer Guide

Information in this document is subject to change without notice. Examples provided are fictitious. No part of this document may be reproduced or transmitted in any form, or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of Actuate Corporation.

© 1995 - 2012 by Actuate Corporation. All rights reserved. Printed in the United States of America.

Contains information proprietary to:

Actuate Corporation, 951 Mariners Island Boulevard, San Mateo, CA 94404

www.actuate.com

www.birt-exchange.com

The software described in this manual is provided by Actuate Corporation under an Actuate License agreement. The software may be used only in accordance with the terms of the agreement. Actuate software products are protected by U.S. and International patents and patents pending. For a current list of patents, please see <http://www.actuate.com/patents>.

Actuate Corporation trademarks and registered trademarks include:

Actuate, ActuateOne, the Actuate logo, Archived Data Analytics, BIRT, BIRT 360, BIRT Data Analyzer, BIRT Performance Analytics, Collaborative Reporting Architecture, e.Analysis, e.Report, e.Reporting, e.Spreadsheet, Encyclopedia, Interactive Viewing, OnPerformance, Performancesoft, Performancesoft Track, Performancesoft Views, Report Encyclopedia, Reportlet, The people behind BIRT, X2BIRT, and XML reports.

Actuate products may contain third-party products or technologies. Third-party trademarks or registered trademarks of their respective owners, companies, or organizations include:

Mark Adler and Jean-loup Gailly (www.zlib.net): zlib. Adobe Systems Incorporated: Flash Player. Apache Software Foundation (www.apache.org): Axis, Axis2, Batik, Batik SVG library, Commons Command Line Interface (CLI), Commons Codec, Derby, Hive driver for Hadoop, Shindig, Struts, Tomcat, Xalan, Xerces, Xerces2 Java Parser, and Xerces-C++ XML Parser. Castor (www.castor.org), ExoLab Project (www.exolab.org), and Intalio, Inc. (www.intalio.org): Castor. Codejock Software: Xtreme Toolkit Pro. Eclipse Foundation, Inc. (www.eclipse.org): Babel, Data Tools Platform (DTP) ODA, Eclipse SDK, Graphics Editor Framework (GEF), Eclipse Modeling Framework (EMF), and Eclipse Web Tools Platform (WTP), licensed under the Eclipse Public License (EPL). Bits Per Second, Ltd. and Graphics Server Technologies, L.P.: Graphics Server. Gargoyle Software Inc.: HtmlUnit, licensed under Apache License Version 2.0. GNU Project: GNU Regular Expression, licensed under the GNU Lesser General Public License (LGPLv3). HighSlide: HighCharts. IDAutomation.com, Inc.: IDAutomation. Jason Hsueh and Kenton Varda (code.google.com): Protocole Buffer. IDR solutions Ltd.: JBIG2, licensed under the BSD license. ImageMagick Studio LLC.: ImageMagick. InfoSoft Global (P) Ltd.: FusionCharts, FusionMaps, FusionWidgets, PowerCharts. Matt Inger (sourceforge.net): Ant-Contrib, licensed under Apache License Version 2.0. Matt Ingenthron, Eric D. Lambert, and Dustin Sallings (code.google.com): Spymemcached, licensed under the MIT OSI License. International Components for Unicode (ICU): ICU library. jQuery: jQuery, licensed under the MIT License. Yuri Kanivets (code.google.com): Android Wheel gadget, licensed under the Apache Public License (APL). KL Group, Inc.: XRT Graph, licensed under XRT for Motif Binary License Agreement. LEAD Technologies, Inc.: LEADTOOLS. Bruno Lowagie and Paulo Soares: iText, licensed under the Mozilla Public License (MPL). Microsoft Corporation (Microsoft Developer Network): CompoundDocument Library. Mozilla: Mozilla XML Parser, licensed under the Mozilla Public License (MPL). MySQL Americas, Inc.: MySQL Connector. Netscape Communications Corporation, Inc.: Rhino, licensed under the Netscape Public License (NPL). OOPS Consultancy: XMLTask, licensed under the Apache License, Version 2.0. Oracle Corporation: Berkeley DB, Java Advanced Imaging, JAXB, JDK, Jstl. PostgreSQL Global Development Group: pgAdmin, PostgreSQL, PostgreSQL JDBC driver. Progress Software Corporation: DataDirect Connect XE for JDBC Salesforce, DataDirect JDBC, DataDirect ODBC. Rogue Wave Software, Inc.: Rogue Wave Library SourcePro Core, tools.h++. Sam Stephenson (prototype.conio.net): prototype.js, licensed under the MIT license. Sencha Inc.: Ext JS. ThimbleWare, Inc.: JMemcached, licensed under the Apache Public License (APL). World Wide Web Consortium (W3C)(MIT, ERCIM, Keio): Flute, JTIty, Simple API for CSS. XFree86 Project, Inc.: (www.xfree86.org): xvfb. ZXing authors (code.google.com): ZXing, licensed under the Apache Public License (APL).

All other brand or product names are trademarks or registered trademarks of their respective owners, companies, or organizations.

Document No. 120201-2-640301 October 22, 2012

Contents

About <i>Information Console Developer Guide</i>	ix
---------------------------------------------------------------	-----------

Part 1

Customizing Actuate Information Console

Chapter 1

Introducing Actuate Information Console	3
------------------------------------------------------	----------

About Actuate Information Console	4
Setting up Actuate Information Console	5
Generating a web archive (WAR) for installation	5
Understanding Actuate Information Console load balancing	6
Deploying a load balancer for an Actuate BIRT iServer cluster	7
About using a cluster of application servers	7
About Actuate Information Console architecture	8
Using proxy servers with Actuate Information Console	8
About Actuate Information Console pages	10
Working with Actuate Information Console URIs	11
About Actuate Information Console URIs	11
Using a special character in a URI	12
About UTF-8 encoding	14
About Actuate Information Console functionality levels	14
Customizing functionality levels	16
Customizing functionality level features	18
Preserving functionality levels and features	20
Using Actuate Analytics experience levels	20
Understanding experience levels	20
Customizing experience levels	21

Chapter 2

Creating a custom Information Console web application	29
--------------------------------------------------------------------	-----------

Information Console web application structure and contents	30
Understanding Information Console directory structure	31
Building a custom Information Console context root	35
Activating a new or custom web application	37
Configuring a custom Information Console web application	37
Customizing Information Console configuration	38
Setting the default locale	39
Controlling the Message Distribution service Load Balancing	39
Specifying the default Encyclopedia volume and server	40

Modifying text and messages	42
Customizing Information Console text and messages	42
Customizing Actuate BIRT iServer error messages	45
Customizing an Information Console web application	47
Modifying the landing page	48
Viewing modifications to a custom web application	49
Locating existing pages and linking in new pages	50
Obtaining information about the user and the session	51
Customizing accessible files and page structure using templates	52
Specifying a template and template elements	53
About the dashboard template	55
Changing a template	55
Modifying existing content or creating new content	56
Modifying global style elements	57
Customizing Actuate Information Console using skins	57
Using skins	58
Managing skins using the skin manager	59
Customizing and cloning skins	60
Understanding style definition files	65
Specifying colors and fonts	66
Customizing page styles for BIRT Studio	67
Modifying graphic images	68

Part 2

Actuate Information Console reference

Chapter 3

Actuate Information Console configuration	73
About Information Console configuration	74
Configuring the Information Console web application	74
Configuring the Information Console using web.xml	74
Configuring Information Console using volumeProfile.xml	80
Using a volume profile defined in volumeProfile.xml	81
Overriding the volume specified in a volume profile	81
Understanding temporary volume profiles	82
Configuring Information Console functionality levels with functionality-level.config	82
Configuring Information Console locales	86
Configuring Information Console time zones	86
Customizing messages and text according to locale	87
Configuring Shindig 2.0 for a WAR or EAR deployment	88
Configuring the connection to iServer	89
Configuring Actuate Analytics	90

Configuring Actuate Analytics Cube Viewer	90
Configuring experience levels for Actuate Analytics Cube Viewer	91
Defining an experience level	93
Adding an experience level to a functionality level	94
Configuring the BIRT Viewer and Interactive Viewer	94
Configuring BIRT Studio	94
Configuring BIRT Data Analyzer	95

Chapter 4

Actuate Information Console URIs 97

Actuate Information Console URIs overview	98
Actuate Information Console URIs quick reference	98
Common URI parameters	101
Information Console Struts actions	102
Actuate Information Console URIs reference	111
about page	114
authenticate page	114
banner page	115
browse file page	116
calendar page	116
channels page	116
completed request page	117
create folder page	117
create query page	118
dashboard page	118
delete file status page	119
delete job page	119
delete status page	119
detail page	120
drop page	122
error page	124
execute page	124
execute query page	125
execute report page	126
general options page	130
get saved search page	131
home page	131
index page	132
license page	135
list page	136
login banner page	139
login page	139
logout page	140

My dashboard page	140
notification page	141
options page	141
output page	143
page not found page	145
parameters page	145
pending page	146
ping page	146
print page	149
privileges page	149
running page	150
save as page	150
schedule page	152
scheduled job page	153
search folders page	154
submit job page	155
submit page	159
view cube page	160
Actuate BIRT Viewer URIs reference	161
Actuate Viewer URIs reference	161
request search page	162
search frame page	162
search report page	162
search toolbar page	164
view default page	165
view frame set page	166
view navigation page	168
view TOC page	169

Chapter 5

Actuate Information Console JavaScript 171

Actuate Information Console JavaScript overview	172
Actuate Information Console JavaScript reference	172

Chapter 6

Actuate Information Console servlets 175

Information Console Java servlets overview	176
About the base servlet	176
Invoking a servlet	176
Information Console Java servlets quick reference	177
Information Console Java servlets reference	177
DownloadFile servlet	177
DownloadSearchResult servlet	178

ExecuteReport servlet	179
GetDynamicData servlet	182
GetReportData servlet	182
GetStaticData servlet	185
Interactive Viewer servlet	186
ViewEmbeddedObject servlet	187
ViewPage servlet	188

Chapter 7

Actuate Information Console custom tags 193

Information Console custom tag overview	194
Information Console custom tags quick reference	194
Information Console custom tag libraries	194
Information Console custom tags	195
Information Console custom tags reference	196
bundle	196
component	198
componentIdentifier	198
componentIdentifierList	199
componentList	199
content	200
copyFileFolder	200
formatDate	201
getFormats	202
getPageCount	203
getReportlet	204
getReportletData	206
getTOC	208
iterator	209
login	210
message	211
searchReport	212
selectUsers	214
string	215
stringList	215
tab	215
tabBegin	216
tabEnd	217
tabMiddle	218
tabMiddleSelected	218
tabPanel	219
tabSeparator	221

Chapter 8	
Actuate Information Console JavaBeans	223
Information Console JavaBeans overview	224
Information Console JavaBeans package reference	224
Information Console JavaBeans class reference	224
Channels	224
Cubes, information objects, and queries	225
Documents	226
General	226
Jobs	227
Skins	227
Users	228
Information Console UserInfoBean class reference	229
Chapter 9	
Using Actuate Information Console security	237
About Actuate Information Console security	238
Protecting corporate data	238
Protecting corporate data using firewalls	238
Protecting corporate data using Network Address Translation	239
Protecting corporate data using proxy servers	239
Understanding the authentication process	239
Creating a custom security adapter	240
Accessing the IPSE Java classes	241
Creating a custom security adapter class	241
Deploying a custom security adapter	242
Understanding the security adapter class	243
Creating an upload security adapter	246
Accessing the necessary Java classes	247
Creating a custom security adapter class	247
Deploying an upload security adapter	248
Understanding the upload security adapter interface	249
Chapter 10	
Customizing Information Console online help	251
About Actuate Information Console online help files	252
Understanding the Information Console help directory structure	252
Understanding a help collection	254
Understanding a document root	255
Understanding context-sensitive help	255
Understanding locale support	256
Using a custom help location	257
Creating a localized help collection	259

Customizing icons, links, and the company logo	261
Changing the corporate logo	261
Changing the corporate logo on the title page	262
Changing the logo in help content pages	263
Changing the additional links footer in help content pages	264
Changing the Google translate element in help content pages	265
Changing icons	266
Changing the browser window title	267
Changing help content	268
Changing existing help content	268
Adding or removing help topics	269
Adding and removing content files	270
Changing the table of contents	271
Changing the index	274
Index	277

About Information Console Developer Guide

Information Console Developer Guide is a guide to designing, deploying, and accessing custom reporting web applications using Actuate Information Console.

Information Console Developer Guide includes the following chapters:

- *About Information Console Developer Guide.* This chapter provides an overview of this guide.
- *Part 1. Customizing Actuate Information Console.* This part describes how to use Information Console and how to customize its appearance and layout.
- *Chapter 1. Introducing Actuate Information Console.* This chapter introduces Actuate Information Console web applications and explains how Information Console works.
- *Chapter 2. Creating a custom Information Console web application.* This chapter explains how to work with Information Console JSP files to design custom reporting web applications.
- *Part 2. Actuate Information Console reference.* This part describes the code components that make up Information Console, such as URIs, JavaScript files, servlets, tags, beans, and security facilities.
- *Chapter 3. Actuate Information Console configuration.* This chapter describes the Information Console configuration files and parameters.
- *Chapter 4. Actuate Information Console URIs.* This chapter describes the Information Console JSPs and URL parameters.
- *Chapter 5. Actuate Information Console JavaScript.* This chapter describes the Information Console JavaScript files.
- *Chapter 6. Actuate Information Console servlets.* This chapter describes the Information Console Java servlets.

- *Chapter 7. Actuate Information Console custom tags.* This chapter describes the Information Console custom tag libraries.
- *Chapter 8. Actuate Information Console JavaBeans.* This chapter lists the Information Console JavaBeans.
- *Chapter 9. Using Actuate Information Console security.* This chapter introduces the Information Console Security Extension (IPSE) and explains how to use it.
- *Chapter 10. Customizing Information Console online help.* This chapter describes how to customize the Information Console online help files.

Part One

Customizing Actuate Information Console

1

Introducing Actuate Information Console

This chapter contains the following topics:

- About Actuate Information Console
- About Actuate Information Console architecture

About Actuate Information Console

Actuate Information Console is a web application that supports accessing and working with report information using a web browser. Web developers and designers use Actuate Information Console's industry-standard technology to design custom e.reporting web applications that meet business information delivery requirements.

Actuate Information Console technology is platform-independent and customizable. By separating user interface design from content generation, Information Console ensures that reporting web application development tasks can proceed simultaneously and independently. You deploy Actuate Information Console on a network with Actuate BIRT iServer. Information Console accesses and stores documents on an Encyclopedia volume managed by iServer. Actuate Information Console technology is also scalable and supports clustering. On a Windows system, the default context root for Information Console is `C:\Program Files\Actuate11\iPortal\iportal` for Information Console installed separately or `C:\Program Files\Actuate11SP4\iServer\servletcontainer\iportal` for Information Console embedded in the BIRT iServer application. On a UNIX-based system, the default context root for Information Console is `$Home/iPortal/iportal` for Information Console installed separately or `$Home/iServer/servletcontainer/iportal` for Information Console embedded in the BIRT iServer application.

Actuate Information Console technology includes the following features:

- JavaServer Pages (JSPs) support creating HTML or XML pages that combine static web page templates with dynamic content.
- Distributing requests to multiple Actuate BIRT iServer machines in an Actuate BIRT iServer System cluster balances server loads.
- Simple Object Access Protocol (SOAP) standards provide plain text transmission of XML using HTTP.
- Actuate Information Delivery API supports direct communication between the pages' custom tags and Actuate BIRT iServer.
- The full range of authentication and authorization functionality that Actuate BIRT iServer provides is available.
- Secure HTTP (HTTPS) supports secure information transfer on the web.
- Licensed options on BIRT iServer provide additional functionality. To use these options on a BIRT iServer System, the BIRT iServer System must be licensed for the options. For example, to use browser-based tools, such as BIRT Interactive Viewer or BIRT Data Analyzer, the BIRT iServer requires the appropriate license options.

The BIRT 360 Option for BIRT iServer is required to use dashboard and gadget files. If these options are not available, users cannot open dashboards or gadgets in Information Console.

Setting up Actuate Information Console

You install Information Console in either of two ways:

- As a separate web application. This method enables native load-balancing for iServer clusters, redundancy to support constant report services over the web, and secure networks using firewalls and proxy servers as described in Chapter 9, “Using Actuate Information Console security.”
- Automatically on the same host with iServer. This method provides reports locally on each iServer machine.

For enterprise architectures, installing Information Console on several web servers is recommended.

To deploy a report to the web, you need:

- An Actuate Information Console installation.
- An application server or JSP or servlet engine such as Actuate embedded servlet engine or IBM WebSphere.
- One or more Actuate designer tools and Actuate BIRT iServer System with Actuate Management Console.
- Actuate BIRT iServer administrator privileges.
- Permission to read, write, and modify operating system directories as necessary. For example, the directory Java uses to hold temporary files is defined by the `java.io.tmpdir` property and is by default the value of the TMP system variable in the Windows environment and `/var/tmp` in the UNIX and Linux environments. Read and write permission must be provided to the application server running Information Console for this directory.

This section discusses deployment concerns that may affect your Information Console installation and how you wish to deploy reports to the web. For more information about installing Information Console, see *Installing BIRT iServer for Windows* or *Installing BIRT iServer for Linux and UNIX*.

Generating a web archive (WAR) for installation

To deploy Information Console on an application server, you can use a WAR file of your Information Console application. Generating Web Archive is a feature of Actuate Information Console that is available to Administrator-level users. This feature creates a WAR file of your entire Actuate Information Console system. Information Console streams the WAR file to your browser. You select a file name and location to save the file. After you customize your system, you can create a

WAR file to deploy the customized Information Console on other machines. The customizations can include any modifications of JavaScript, JavaServer Pages (JSPs) and other web pages, and skins. Later chapters in this book provide detailed information about customizing JavaScript and JSPs.

If Actuate Information Console is deployed as a WAR file, you cannot further customize skins, add pages, or make any other changes that affect the Actuate Information Console file structure in the WAR file. Instead, install Actuate Information Console as a directory structure with the installation wizard on your product CD and make your changes to that installation. Then use Generate Web Archive to create a new WAR file and deploy that WAR file to your application server.

How to customize and deploy Actuate Information Console in a cluster

To customize Actuate Information Console and deploy it to application servers in a clustered environment, use the following general procedure.

- 1 Install Actuate Information Console on one of the machines in your cluster.
- 2 Customize the Actuate Information Console JavaScript, skins, and web pages as desired.
- 3 Open Information Console. On the landing page, choose My Documents.
- 4 Log in as an administrator-level user. On the Information Console banner, choose Customization.
- 5 Choose Generate Web Archive. At the prompt, provide a location for the WAR file. For example, provide the location where your application server accesses WAR files. By default, the name of the WAR file of your customized Actuate Information Console installation is `acweb.war`.
- 6 Deploy the WAR file to each remaining machine in your cluster.

Understanding Actuate Information Console load balancing

Actuate Information Console supports two kinds of load balancing, as illustrated in Figure 1-1, to ensure high availability and to distribute tasks for efficient processing:

- Actuate Message Distribution service (MDS) balances the request load among Actuate BIRT iServer machines in an Actuate BIRT iServer cluster.
The Message Distribution service eliminates the need for a third-party network load balancer in front of the Actuate BIRT iServer tier. Actuate Information Console determines which machines in a cluster have MDS running and detects when the MDS machines go offline. MDS distributes the load among the available servers and does not attempt to send a request to an offline machine.

- Clustered Actuate Information Console machines can use a third-party application to balance the load among the application servers.

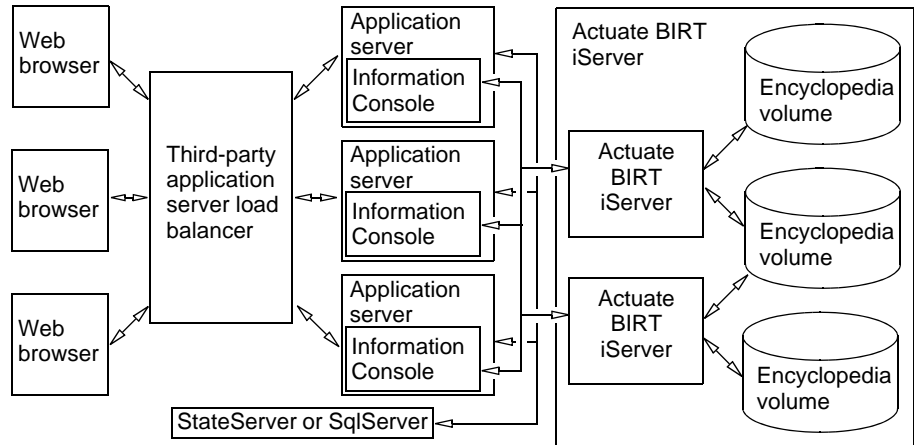


Figure 1-1 Load-balancing architecture for Information Console

Deploying a load balancer for an Actuate BIRT iServer cluster

To deploy a load balancer or proxy layer in front of the Actuate BIRT iServer tier, disable the Actuate load-balancing support by setting the `MDS_ENABLED` configuration parameter to `False` in the `web.xml` Actuate Information Console configuration file.

About using a cluster of application servers

If the application servers running Information Console support session state management, you can configure Actuate Information Console and the application servers to share and maintain a web browsing session state across a cluster of Information Console instances. Configuring the application servers to track the state of each Information Console instance supports reusing authentication information. In other words, you can log in to an Information Console instance and send a request using another Information Console instance without logging in again using the second instance.

If you do not use an application server to track session state information, managing the session state is fast, but you lose a user's state information when you restart Actuate Information Console or your application server.

Sharing session state information takes advantage of the application servers' failover features. If a user is on a cluster application server running Information Console and that application server fails, another application server running Information Console can manage the user's session.

An application server works with one or more database servers to manage session state information. All application servers must have access to the database server to store and retrieve session state information. For specific information about configuring your installation, see your application server documentation.

About Actuate Information Console architecture

This section describes the general operation, authentication, and structure of Information Console as a web application.

The Actuate Information Console architecture is illustrated in Figure 1-2.

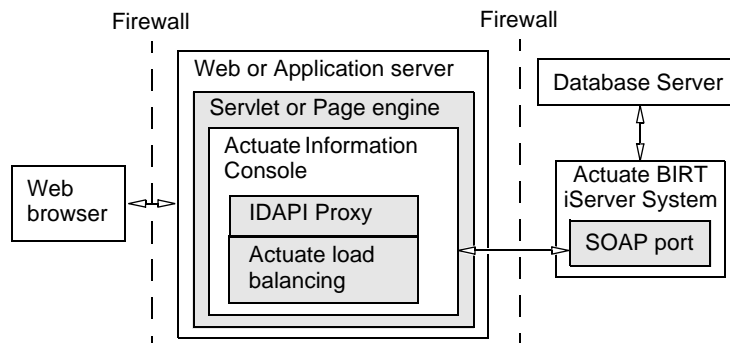


Figure 1-2 Actuate Information Console architecture overview

A user submits a request by choosing a link on a web page that specifies an Actuate Information Console URI. As shown in Figure 1-2, the web or application server receives the URI as an HTTP request and passes the request to the servlet or page engine. The engine invokes Actuate Information Console, interprets the URI, and communicates with the Actuate BIRT iServer using the Actuate Information Delivery API (IDAPI). The IDAPI manages the request and returns the results to Actuate Information Console and the servlet or page engine. The web server returns the results to the web browser. Then, the web browser displays the results for the user.

Actuate Information Console manages requests as part of a JSP engine within a web or application server. There is no default user interface for the engine. On a Windows system, Actuate Information Console installation places an Actuate Information Console link on the Start menu.

Using proxy servers with Actuate Information Console

When setting up a proxy server with Actuate Information Console, there are steps you must take if your internal application server port is protected by a firewall. In this situation, when the proxy server changes the URL to point to the new

context's port, that port is unavailable due to the firewall. The usual solution is to configure a reverse proxy, but if you are using multiple proxies and a reverse proxy is not practical for your installation, Actuate Information Console can perform the redirection.

To redirect a page without using a reverse proxy, Actuate Information Console forwards the URL to redirect to the `processRedirect.jsp` page and updates the browser's location bar accordingly. This action processes on the client. The browser takes the current URL location and updates the rest of the URI using the redirected URL. You must also set the `ENABLE_CLIENT_SIDE_REDIRECT` configuration parameter to `True` and modify the `redirect` attributes in the `<context root>/WEB-INF/struts-config.xml` file. The necessary modifications are included in the file. You just need to comment out the lines that have the `redirect` attribute set to `True` and uncomment the lines that forward to the `processRedirect.jsp` page.

For example, the following code is the `struts-config.xml` entry for the login action:

```
<!-- Process a user login -->
<action path="/login" name="loginForm"
    scope="request" input="/iportal/activePortal/private/login.jsp"
    type="com.actuate.activeportal.actions.AcLoginAction"
    validate="false">
    <forward name="loginform"
        path="/iportal/activePortal/private/login.jsp" />
<!--
<forward name="success" path="/iportal/activePortal/private/common
    /processredirect.jsp?redirectPath=/getfolderitems.do" />
-->
<forward name="success" path="/getfolderitems.do"
    redirect="true" />
<forward name="dashboard" path="/dashboard" redirect="true" />
<forward name="ajcLogin" path="/ajclanding.jsp" redirect="true" />
<forward name="landing" path="/landing.jsp" redirect="false" />
</action>
```

By default the forward statement for success points to `getfolderitems.do` with the `redirect` attribute set to `True`. This code instructs the application server to send a redirect with the `getfolderitems.do` URL when the user logs in.

From behind a firewall and proxy, this redirect method fails because the redirect sent by the application server points to the application server port instead of the firewall and proxy port. For success, comment out the line having `redirect="true"`. Uncomment the line that points to `processRedirect.jsp`. The following code shows the updated entry in `struts-config.xml`:

```
<!-- Process a user login -->
<action path="/login" name="loginForm"
    scope="request" input="/iportal/activePortal/private/login.jsp"
    type="com.actuate.activeportal.actions.AcLoginAction"
    validate="false">
```

```

<forward name="loginform"
    path="/iportal/activePortal/private/login.jsp" />
<forward name="success" path="/iportal/activePortal/private/common
    /processredirect.jsp?redirectPath=/getfolderitems.do" />
<!--
<forward name="success" path="/getfolderitems.do"
    redirect="true" />
-->
<forward name="dashboard" path="/dashboard" redirect="true" />
<forward name="ajcLogin" path="/ajclanding.jsp" redirect="true" />
<forward name="landing" path="/landing.jsp" redirect="false" />
</action>

```

This change needs to be made for all the actions in struts-config.xml that send a redirect to the browser.

About Actuate Information Console pages

Actuate Information Console uses JSPs to generate web pages dynamically before sending them to a web browser. These JSPs use custom tags, custom classes, and JavaScript to generate dynamic web page content. The JavaScript, classes, and tags provide access to other pages, JavaBeans, and Java classes. For example, application logic in Actuate Information Console can reside on the web server in a JavaBean.

Web browsers can request a JSP with parameters as a web resource. The first time a web browser requests a page, the page is compiled into a servlet. Servlets are Java programs that run as part of a network service such as a web server. Once a page is compiled, the web server can fulfill subsequent requests quickly, provided that the page source is unchanged since the last request.

The dashboards servlet and JSPs support the dashboards and gadgets interface for Information Console. The dashboard pages reside in <context root>\dashboard\jsp. To provide dashboard access, enable the BIRT 360 license option.

The channels JSPs and custom tags support viewing reports submitted to channels. The channels pages reside in <context root>\iportal\activePortal\private\channels. Users access channels by clicking Channel in the sidebar.

The filesfolders JSPs and custom tags support accessing repository files and folders. These JSPs and custom tags reside in <context root>\iportal\activePortal\private\filesfolders.

The submit request JSPs and custom tags support submitting new jobs. The submit request JSPs reside in <context root>\iportal\activePortal\private\newrequest. For specific information about running jobs using Actuate Information Console, see *Using Information Console*.

The options JSPs and custom tags support managing user option settings. The options pages reside in `<context root>\portal\activePortal\private\options`.

The viewing JSPs and custom tags support the following functionality, depending on the report type:

- Searching report data
- Using a table of contents to navigate through a report
- Paginating or not paginating a report
- Fetching reports in supported formats

For specific information about viewing reports using Actuate Information Console, see *Using Information Console*.

Use the default pages, customize the pages, or create entirely new pages to deploy your reporting web application.

Working with Actuate Information Console URIs

Actuate Information Console Uniform Resource Identifiers (URIs) convey user requests to the Actuate BIRT iServer System. URIs access functionality including generating and storing reports, managing volume contents, and viewing reports.

About Actuate Information Console URIs

Actuate Information Console URIs consist of the context root and port of the web server where you install and deploy the JSPs or servlets. Actuate Information Console URIs have the following syntax:

```
http://<web server>:<port>/portal/<path><page>.<type>
[?<parameter=value>{&<parameter=value>}]
```

- `<web server>` is the name of the machine running the application server or servlet engine. You can use `localhost` as a trusted application's machine name if your local machine is running the server.
- `<port>` is the port on which you access the application server or page or servlet engine. The default port for Information Console installed separately is 8700, while the BIRT iServer embedded version uses 8900 by default.
- `portal` is the default context root for accessing the Actuate Information Console pages.
- `<path>` is the directory containing the page to invoke.
- `<page>` is the name of the page or method.
- `<type>` is `jsp` or `do`.
- `<parameter=value>` specifies the parameters and values that the page requires.

For example, to view the login page, Actuate Information Console uses a URI with the following format:

```
http://<web server>:<port>/iportal  
/login.jsp?TargetPage=<folder/file>
```

- `iportal/login.jsp` is the JSP that provides default login functionality for Information Console.
- `TargetPage` is the `viewframeset.jsp` parameter that specifies the page to direct the user to after the login completes.
- `<folder/file>` is the complete pathname for the file that the client opens after the login completes.

Using a special character in a URI

Actuate Information Console URIs use encoding for characters that a browser can misinterpret. The following example uses hexadecimal encoding in the Information Console URI to display the report, `Msbargph.roi`, from an Encyclopedia volume:

```
http://infoconsole:8900/iportal/activePortal/viewer  
/viewframeset.jsp?name=%2fmsbargph%2eroi%3b1&__vp=server1
```

You do not have to use hexadecimal encoding in all circumstances. Use the encoding only when the possibility of misinterpreting a character exists. The following unencoded URI displays the same report as the preceding URI:

```
http://infoconsole:8900/iportal/activePortal/viewer  
/viewframeset.jsp?name=\msbargph.roi;1&__vp=server1
```

Always encode characters that have a specific meaning in a URI when you use them in other ways. Table 1-1 describes the available character substitutions. An ampersand introduces a parameter in a URI, so you must encode an ampersand that appears in a value string. For example, use:

```
&company=AT%26T
```

instead of:

```
&company=AT&T
```

Table 1-1 Encoding sequences for use in URIs

Character	Encoded substitution
ampersand (&)	%26
asterisk (*)	%2a
at (@)	%40
backslash (\)	%5c
colon (:)	%3a

Table 1-1 Encoding sequences for use in URIs

Character	Encoded substitution
comma (,)	%2c
dollar sign (\$)	%24
double quote (")	%22
equal (=)	%3d
exclamation (!)	%21
forward slash (/)	%2f
greater than (>)	%3e
less than (<)	%3c
number sign (#)	%23
percent (%)	%25
period (.)	%2e
plus (+)	%2b
question mark (?)	%3f
semicolon (;)	%3b
space ()	%20
underscore (_)	%5f

If you customize Actuate Information Console by writing code that creates URI parameters, encode the entire parameter value string with the `encode()` method. The `encode()` method is included in `encoder.js`, which is provided in the Actuate Information Console `<context root>/js` directory. The following example encodes the folder name `/Training/Sub Folder` before executing the `getFolderItems` action:

```
<%-- Import the StaticFuncs class. --%>
<%@ page import="com.actuate.reportcast.utils.*" %>

<%
    String url =
        "http://localhost:8900/iportal/getfolderitems.do?folder=" +
        StaticFuncs.encode("/Training/Sub Folder");
    response.sendRedirect(url);
%>
```

The `encode()` method converts the folder parameter value from:

`/Training/Sub Folder`

to:

`%2fTraining%2fSub%20Folder`

About UTF-8 encoding

All communication between Information Console and BIRT iServer uses UTF-8 encoding. UTF-8 encoding is the default encoding that web browsers support. For 8-bit (single-byte) characters, UTF-8 content appears the same as ANSI content. However, if extended characters are used (typically for languages that require large character sets), UTF-8 encodes these characters with two or more bytes.

UTF-8 encoding support is encoded for all Information Console web pages. When customizing these pages or adding customized web pages to an Information Console web application, provide UTF-8 encoding support using the following code:

```
<META  
  HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
```

About Actuate Information Console functionality levels

Actuate Information Console provides functionality levels that control which features are available to a user. By default, each user can access all of the functionality level features. To restrict access to features for user groups, the Actuate Information Console administrator can modify functionality levels and add additional levels by editing the configuration file. The standard location for the Actuate Information Console configuration file is <context root>\WEB-INF\functionality-level.config.

Functionality-level.config has several functionality levels mapped to security roles, much like privileges, in comments. Table 1-2 shows the supplied functionality levels and their corresponding security roles.

Table 1-2 Functionality levels mapping to security roles

Functionality level	Security role
Basic	All—default access
Intermediate	Active Portal Intermediate
Advanced	Active Portal Advanced
Administrator	Active Portal Administrator

When uncommenting existing security roles or creating new security roles, make sure that any roles specified in the configuration file also exist in the Encyclopedia volume. Because all users automatically belong to the All security role, all users receive the functionality associated with the Basic or the Open functionality level plus the functionality associated with any other roles they have. When restricting access to features, remove the feature from the Open functionality level or comment out the Open level completely and use the Basic functionality level.

Understanding the provided functionality levels

When the comment tags are removed, the provided functionality levels give the following access. Users with the Basic level can perform the following tasks:

- Access Documents, My Jobs, and Channels
- Delete their own files

Basic level users cannot perform any other modifications. The default banner for the Basic level looks like the one in Figure 1-3.

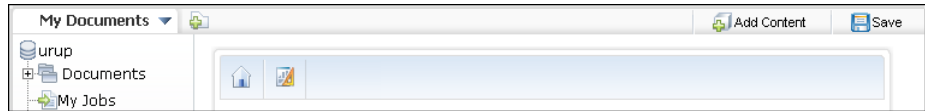


Figure 1-3 Banner menu for a basic level Actuate Information Console user

Users at the Intermediate level have all the Basic level access, and can also perform the following tasks:

- Search documents.
- Create their own job notifications with attachments.
- Subscribe to channels.
- Upload and download files.
- Use the interactive viewer, if this option is licensed.

Users at the Advanced level have all the Intermediate level access, plus they can perform the following tasks:

- Create and delete folders.
- Share files and folders.
- Set job priority.

The default banner for the Intermediate and Advanced levels adds a Search link and looks like the banner in Figure 1-4.

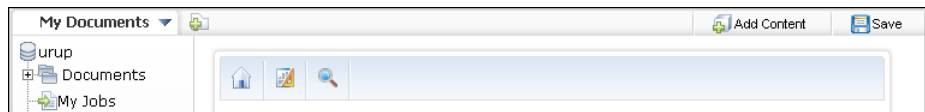


Figure 1-4 Banner menu for advanced level Actuate Information Console user

Users at the Administrator level can perform all Advanced level tasks and can also clone and customize Actuate Information Console skins. The default banner for the Administrator level adds a Customization link, activates the add content function, and looks like the banner in Figure 1-5.

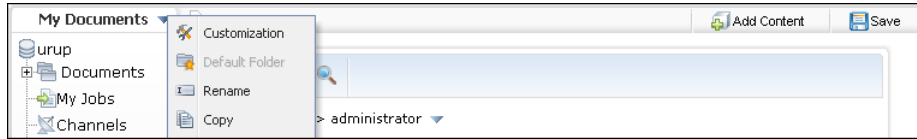


Figure 1-5 Banner menu for an administrator Actuate Information Console user

Use Actuate Management Console to associate the levels with users in the Encyclopedia volume by assigning the appropriate roles to each user.

Customizing functionality levels

Customize or add functionality levels by modifying or creating a level definition in `functionality-level.config`. A functionality level definition consists of five parts:

- **Level name**
The level name must be a unique alphanumeric string, enclosed within `<Name>` and `</Name>` tags.
- **Matching security role**
The name of the security role that corresponds to the functionality level. Both the security level and the functionality level must exist before the functionality level can be assigned to a user. Enclose the role name with `<Role>` and `</Role>` tags.
- **Available features**
Table 1-3 describes the five available features.

Table 1-3 Features for functionality levels

Feature	Description
Channels	Provides access to channels
Customization	Provides access to skin customization
Documents	Provides access to files and folders
Jobs	Allows submitting and accessing jobs
Mobile	Provides access to BIRT mobile viewing
Search	Provides access to the file search facility

Features are specified one per line and are enclosed within `<FeatureID>` and `</FeatureID>` tags. When a feature is omitted from a functionality level, the corresponding side menu or banner item is hidden to anyone assigned that functionality level. For example, the Search feature is not provided in the Basic functionality level, so the Search link does not appear for users with the Basic functionality level.

- Available subfeatures

Subfeatures correspond to actions that you can perform through Actuate Information Console. Most subfeatures are associated with a feature. A subfeature cannot be included in a functionality level if its corresponding feature is not included. The subfeatures are described in Table 1-4.

Table 1-4 Subfeatures for functionality levels

Subfeature	Feature	Description
AddFile	Documents	Permits adding files when the user has the appropriate privileges
AdvancedData	NA	Permits the modifying and synchronizing of data sets in BIRT Studio
CreateFolder	Documents	Permits creating folders when the user has the appropriate privileges
Dashboard BusinessUser	NA	Permits use of dashboards
Dashboard Developer	NA	Permits design and administration of dashboards
DeleteFile	Documents	Permits deleting files when the user has the appropriate privileges
DeleteFolder	Documents	Permits deleting folders when the user has the appropriate privileges
DownloadFile	Documents	Permits downloading files when the user has the appropriate privileges
InteractiveViewing	NA	Permits opening Interactive Viewer
<i>(continues)</i>		
JobPriority	Jobs	Permits setting job priority, up to the user's maximum job priority
SelfNotification WithAttachment	Jobs	Activates e-mail notification for successful jobs
ShareDashboard	NA	Permits sharing dashboards when the user has the appropriate privileges
ShareFile	Documents	Permits sharing files when the user has the appropriate privileges
SubscribeChannel	Channels	Permits subscribing to channels

Subfeatures are specified one per line, enclosed within <SubfeatureID> and </SubfeatureID> tags.

- Available Actuate Analytics user experience levels
Users can select their own Actuate Analytics user experience level on the Actuate Information Console Options page from the levels listed here. The following Actuate Analytics user experience levels are available at this functionality level:
 - Novice
 - Standard
 - Advanced

The following code shows a sample functionality level entry:

```
<Level>
  <Name>ViewAndSearch</Name>
  <Role>All</Role>
  <FeatureID>Jobs</FeatureID>
  <FeatureID>Documents</FeatureID>
  <FeatureID>Search</FeatureID>
  <SubfeatureID>ShareFile</SubfeatureID>
  <SubfeatureID>DeleteFile</SubfeatureID>
  <AnalyticsExperienceLevel>Novice</AnalyticsExperienceLevel>
  <AnalyticsExperienceLevel>Standard</AnalyticsExperienceLevel>
  <AnalyticsExperienceLevel>Advanced</AnalyticsExperienceLevel>
</Level>
```

The level is named ViewAndSearch and is available to all security roles. Users with ViewAndSearch functionality can run jobs, access documents, and search for files. In addition, they can share and delete their own files, and set their Actuate Analytics experience level to any of the available levels.

Customizing functionality level features

Customize functionality level features by modifying the action they perform and the graphic image they use. Features are defined in the functionality-level.config file. A feature definition consists of up to five parts:

- Feature ID
This is the feature name and must be a unique alphanumeric string, enclosed within <ID> and </ID> tags. This value is used as the feature name in functionality level definitions. Do not change this value, because the IDs are used in the Actuate Information Console code to identify the features.
- Label key
This key is used in the Actuate Information Console resource files. These files have names of the format, ActivePortalResources_<locale>.properties. The files are located in <context root>\WEB-INF\lib\resources.jar. If this file does not contain a resource file for a locale, the resource file, ActivePortalResources.properties, for the default locale, en_US, is used. The

key provides for proper translation in the resource file so that the hyperlink text for the feature is displayed using the current locale. Keys are enclosed within <Labelkey> and </Labelkey> tags. Do not change the key values or the resource string substitution fails.

- **Link**

This link is the target URI of the label key hyperlink, which is typically to the page that corresponds to the feature. Table 1-5 shows the targets for each feature. Links are enclosed within <Link> and </Link> tags. Change the link target for the feature by replacing the default page or action name.

Table 1-5 Actuate Information Console targets for features

Feature	Actuate Information Console target
Documents	\getfolderitems.do
Jobs	\selectjobs.do
Channels	\selectchannels.do
Search	\searchfiles.do
Customization	\customize.do

- **Large icon and Small icon**

These optional icons are displayed together with the link, depending on the skin. For example, the Classic skin displays the large icons, the Treeview skin uses the small icons, and the Tabbed skin does not use these icons at all. Table 1-6 shows features and their icons. Large icons are 32 pixels square. Their file names are relative to the context root and are enclosed within <LargeIcon> and </LargeIcon> tags. Small icons are 16 pixels square. Their file names are relative to the context root and are enclosed within <SmallIcon> and </SmallIcon> tags. Replace these file names with the names of your own icons to customize your skin's appearance.

Table 1-6 Icons for features

Feature	SmallIcon	LargeIcon
Documents	\images\filesfoldersicon16x16.gif	\imagesfilesfoldersicon.gif
Jobs	\images\requestsicon16x16.gif	\imagesrequestsicon.gif
Channels	\images\channelsicon16x16.gif	\imageschannelsicon.gif

The following example shows a sample definition for the Channels feature. This example specifies custom large and small icons. The Classic and Treeview skins, and any skins cloned from them, use these new images for the channel icon.

```

<Feature>
  <ID>Channels</ID>
  <Labelkey>SBAR_CHANNELLS</Labelkey>
  <Link>/selectchannels.do</Link>
  <SmallIcon>/images/customIcon16x16.gif</SmallIcon>
  <LargeIcon>/images/customIcon32x32.gif</LargeIcon>
</Feature>

```

Preserving functionality levels and features

The functionality-levels.config file is overwritten during upgrade installations. This change ensures that new levels, features, and subfeatures are available to you with your new Actuate Information Console installation. If you have modified your existing functionality-level.config file, make a backup of the changes before the upgrade. Use the backed-up file to access your changes and merge them into the new functionality-level.config file.

Using Actuate Analytics experience levels

If you have purchased the Actuate Analytics Option, additional customization features are available. The Actuate Analytics Cube Viewer uses experience levels to disable Cube Viewer features based on the experience level that a user chooses.

Understanding experience levels

The following list shows the three standard Actuate Analytics experience levels:

- Novice
- Standard
- Advanced

The Actuate Information Console Administrator can edit the experience.levels configuration file that defines the levels to modify the levels and add additional level definitions. The location for the Actuate Information Console configuration file is <context root>\WEB-INF. The user can also choose their default experience level on the Actuate Information Console Options—General page. Actuate Information Console stores the choice as part of the user's profile. If the experience.levels file is missing, all functionality becomes available to all users.

Every Actuate Information Console functionality level includes a list of Actuate Analytics Cube Viewer experience level names. The experience level names must match the experience level configuration names in the experience.levels file. This list controls the experience levels available to the user for that functionality level. The user can choose among the experience levels available when viewing a cube.

Customizing experience levels

As the Actuate Information Console Administrator, you can create and modify experience levels. You modify a level by adding or removing HIDEITEM entries to hide or not hide a part of the Cube Viewer user interface. The following tables, which are organized by their user interface component, describe these entries. The Experience level column shows the most restricted experience level that displays the user interface component. For example, Standard indicates that both Standard and Advanced show the element but that Novice does not. An experience level of None means that none of the supplied levels show that element.

Display of the Cube Viewer horizontal bars are controlled with the elements in Table 1-7.

Table 1-7 Tags that control Cube Viewer horizontal bars

HIDEITEM tag keyword	Functionality	Experience level
ENTIRE_BANNER	Banner	Novice
ENTIRE_DIMENSIONBAR	Categories bar	Novice
ENTIRE_REPORTBAR	Report bar	None
ENTIRE_TOOLBAR	Toolbar	Novice
ENTIRE_TITLEBAR	Title bar	Novice

Cube Viewer toolbar buttons are controlled with the elements in Table 1-8.

Table 1-8 Tags that control Cube Viewer toolbar buttons

HIDEITEM tag keyword	Functionality	Experience level
ABOUT_TB	About	Novice
methodS_TB	Calculate	Advanced
COLLABORATE_TB	Collaborate	Novice
EXCEPTION_TB	Exception highlighting	Advanced
HELP_TB	Help	Novice
HOME_TB	Home	None
HORIZONTAL_BAR_CHART_TB	Horizontal bar chart	Novice
LINE_GRAPH_TB	Line graph	Standard
PIE_CHART_TB	Pie chart	Novice
PREFERENCES_TB	Preferences	Novice
PRINT_TB	Print	Novice

(continues)

Table 1-8 Tags that control Cube Viewer toolbar buttons (continued)

HIDEITEM tag keyword	Functionality	Experience level
SAVE_AS_TB	Save	Novice
EXPORT_TB	Save as Microsoft Excel	Advanced
EXPORTDOC_TB	Save as Microsoft Word	Advanced
TABLE_VIEW_TB	Table view	Novice
FIT_TO_PAGE_TB	Vertical or horizontal fit to page	Standard
VERTICAL_BAR_CHART_TB	Vertical bar chart	Novice
VIEW_TB	Presentation or analysis view	Standard
VIEWS_TB	Reports combo box	Standard
EDIT_UNDO_REDO_TB	Undo/Redo	Novice

Some toolbar buttons have their own menus. The menu items are controlled with the elements in Table 1-9, grouped by button.

Table 1-9 Tags that control Cube Viewer submenu items

HIDEITEM tag keyword	Functionality	Experience level
Save As submenu of the Save button		
SAVEAS_PDF	Adobe PDF (.pdf)	Novice
SAVEAS_SYLK	Microsoft Excel (.xls)	Novice
SAVEAS_RTF	Microsoft Word (.doc)	Novice
SAVEAS_CSV	Text (.txt comma separated)	Novice
SAVEAS_TSV	Text (.txt tab separated)	Novice
SAVE_SESSION_LOCALLY	Work offline	Standard
Collaborate button menu items		
COLLABORATE_MENU_ITEM	Collaborate	Novice
SEND_REPORT_MENU_ITEM	Send report	Novice
Preference button menu items		
COLUMNS_MENUITEM	Columns	Novice
GENERAL_MENUITEM	General	Novice
ROWS_MENUITEM	Rows	Novice

Table 1-9 Tags that control Cube Viewer submenu items

HIDEITEM tag keyword	Functionality	Experience level
Graphics submenu of Preference button		
BAR_MENUITEM	Bar	Novice
EXPERIENCE_LEVELS_ CASCADE_MENUITEM	Experience levels	Novice
LINE_MENUITEM	Line	Novice
PIE_MENUITEM	Pie	Novice
Calculate button menu items		
AVERAGE	Average	Advanced
DIFFERENCE	Difference	Advanced
GROWTH_PERCENT	%Growth	Advanced
INTERSECT	Intersection (AND)	Advanced
MAX	Maximum value	Advanced
MIN	Minimum value	Advanced
PERFORMANCE_INDEX	Performance index	None
RATIO	Ratio	Advanced
RELATIVE_TIME_PERIOD	Relative time periods	Advanced
SUM	Sum	Advanced
PERCENTAGE_OF_TOTAL_ MENU_ITEM	% of total	Advanced
UNION	Union (OR)	Advanced
Reports button menu items		
HOME	Home	Standard
ADMIN_LOG_ON	Log on as view administrator	Advanced
REPORTS_BOX	Save	Novice
ADD_TITLE	Save as	Standard

The menu items shown after a right-click on a table view, bar chart view, or line chart view are controlled with the elements that are shown in Table 1-10.

Table 1-10 Tags that control Cube Viewer context menu items

HIDEITEM tag keyword	Functionality	Experience level
Right-click on table view		
COLUMN_PERCENTAGES	% of column total	Advanced
GLOBAL_PERCENTAGES	% of grand total	Advanced
ROW_PERCENTAGES	% of row total	Advanced
DRILL_UP	Collapse	Novice
VIEW_SOURCE	Drill through to details	Advanced
DRILL_DOWN	Expand	Novice
EXPORT_SOURCE_TO_SPREADSHEET	Export details to spreadsheet	Advanced
FORMAT_SCALE	Format scale...	Standard
GENERAL_PROPERTIES	Preferences	Standard
VIEW_NUMERIC_DATA	Show cell value with calculation results	Novice
Right-click on a bar in the bar view		
HISTOGRAM_PROPERTIES	Bar chart preferences	Standard
Right-click on the line chart view		
LINE_PROPERTIES	Line preferences	Standard

The menu items shown after right-click on various axis components are controlled with the elements shown in Table 1-11.

Table 1-11 Tags that control Cube Viewer axis context menu items

HIDEITEM tag keyword	Functionality	Experience level
Right-click on the column axis header		
COLUMN_PROPERTIES	Column preferences	Standard
Right-click on the row axis header		
ROW_PROPERTIES	Row preferences	Standard
Right-click on axes, submenu under sort		
CUSTOM_SORT	Custom...	Standard
DESCENDING	Highest to lowest	Novice
ASCENDING	Lowest to highest	Novice
TOP_10	Show highest ten	Novice
BOTTOM_10	Show lowest ten	Novice

Table 1-11 Tags that control Cube Viewer axis context menu items (continued)

HIDEITEM tag keyword	Functionality	Experience level
Right-click on the axes		
APPLY_FILTER	Apply or cancel filter...	Standard
DRILL_UP	Collapse	Novice
Right-click on the axis header		
FILTER_MENU_ITEM	Filter	Standard
Right-click on axis		
EDIT_method	Edit calculation	Standard
DRILL_DOWN_INTO	Expand into	Novice
DRILL_TO_LEVEL	Expand to level	Standard
RELATIVE_DATE_FILTER_MENU_ITEM	Filter by relative time periods...	Advanced
HIDE_SELECTED_UNSELECTED	Hide selected or unselected subcategories	Standard
HIDE_ROWS_COLUMNS_WITH_NO_DATA_IN	Hide rows and columns with no data	Standard
HIDE_ROWS_COLUMNS_WITH_ZEROS_IN	Hide rows and columns with zeros	Standard
SHOW_ALL_CATEGORIES	Cancel hiding	Standard
COLUMN_TOTALS	Show column totals as	Standard
ROW_TOTALS	Show row totals as	Standard
LABEL_STYLE_CASCADE_MENU_ITEM	Show labels as	Standard
Right-click on axis header		
SEARCH_MENU_ITEM	Search...	Standard
PIVOT_AXES	Swap rows and columns	Novice
Right-click on pie view		
LIMIT_SLICES_BY_THRESHOLD	Limit slices by threshold	Standard
LIMIT_SLICES_BY_VIEWPORT	Limit slices by view port	Standard
PIE_PROPERTIES	Pie preferences	Standard
SHOW_ALL_SLICES	Show all slices	Standard
SHOW_NEXT_LARGEST_CATEGORY	Show next largest slice	Standard

(continues)

Table 1-11 Tags that control Cube Viewer axis context menu items (continued)

HIDEITEM tag keyword	Functionality	Experience level
Right-click on row or column totals right-click menu item		
LEADING_CASCADE_ITEM	Leading	Novice
NONE_CASCADE_ITEM	None	Novice
TRAILING_CASCADE_ITEM	Trailing	Novice

When modifying elements in `experience.levels`, do not remove the following entries:

```
<HIDEITEM>EDIT_TITLE</HIDEITEM>
<HIDEITEM>ADMIN_LOG_ON</HIDEITEM>
<HIDEITEM>VIEW_SOURCE</HIDEITEM>
```

These features are not supported in Actuate Analytics, and these elements must appear under every `<EXPERIENCE_LEVEL>` element. The `NUMBER_OF_LEVELS` element value must correspond to the number of experience levels defined in the file. The `DEFAULT_EXPERIENCE_LEVEL` element value specifies the default experience level to use if no level is specified for a user. For information about using experience level items in the Cube Viewer, see *Working with Cube Reports using Actuate Analytics Option*.

How to add an experience level

- 1 Using a text editor that supports UTF-8 encoding, open `experience.levels`. In some system configurations, this file does not already exist in the `WEB-INF` directory for your application. Some editors, such as Microsoft Notepad, add a marker string to the file to identify the UTF-8 encoding. Do not use an editor that adds hidden information to the file.

- 2 Find the `<NUMBER_OF_LEVELS>` tag and increase the number of levels by one. There are three levels in the standard `experience.levels`, so set the new value to 4:

```
<NUMBER_OF_LEVELS>4</NUMBER_OF_LEVELS>
```

- 3 Find the last `</EXPERIENCE_LEVEL>` tag

- 1 Insert the following code after the `</EXPERIENCE_LEVEL>` tag and before the `</EXPERIENCE_LEVELS>` tag:

```
<EXPERIENCE_LEVEL>
  <SKIN_NAME>SampleLevel</SKIN_NAME>
```

The skin name is used by Actuate Information Console on the General—Options page.

2 Insert code in one of the following formats for the display name that Actuate Analytics Cube Viewer uses:

- ❑ To use a static value for the display name:

```
</DISPLAY_NAME>
  <LOCALE_ID>en_US</LOCALE_ID>
  <NAME>Sample</NAME>
</DISPLAY_NAME>
```

- ❑ To use a resource key to access the display name in the appropriate locale:

```
<DISPLAY_NAME>SAMPLE_EXP_LEVEL_ID<DISPLAY_NAME>
```

3 Insert the following code after the `<DISPLAY_NAME>` tag line:

```
<HIDEITEM>HOME_TB</HIDEITEM>
<HIDEITEM>EDIT_TITLE</HIDEITEM>
<HIDEITEM>ADMIN_LOG_ON</HIDEITEM>
<HIDEITEM>VIEW_SOURCE</HIDEITEM>
</EXPERIENCE_LEVEL>
```

4 Save and close experience.levels.

5 Using a text editor, open functionality-level.config.

6 Add the following line to the list of Analytics experience levels in every level:

```
<AnalyticsExperienceLevel>Sample</AnalyticsExperienceLevel>
```

For example, the Basic level should look like the following code:

```
<Level>
  <Name>Basic</Name>
  <Role>All</Role>
  <FeatureID>Jobs</FeatureID>
  <FeatureID>Documents</FeatureID>
  <FeatureID>Channels</FeatureID>
  <SubfeatureID>DeleteFile</SubfeatureID>
  <SubfeatureID>InteractiveViewing</SubfeatureID>
  <AnalyticsExperienceLevel>Novice</AnalyticsExperienceLevel>
  <AnalyticsExperienceLevel>Standard</AnalyticsExperienceLevel>
  <AnalyticsExperienceLevel>Advanced</AnalyticsExperienceLevel>
  <AnalyticsExperienceLevel>Sample</AnalyticsExperienceLevel>
</Level>
```

7 Save and close functionality-level.config.

- 8 To apply these configuration changes, restart your application server or JSP engine. For example, to restart Information Console's embedded servlet engine on a Windows XP system, perform the following steps:
 - 1 From the Windows Start menu, choose Settings→Control Panel.
 - 2 Choose Administrative Tools.
 - 3 Choose Services.
 - 4 On Services, select Actuate 11 Apache Tomcat for Information Console.
 - 5 From the menu, choose Action→Restart.
 - 6 Close Services.

How to select a new experience level

- 1 Choose Start→Programs→Actuate 11→Information Console.
- 2 Log in to Actuate Information Console.
- 3 In Documents, choose Options.
- 4 In General, select the Analytics Experience Level and view the levels that appear in the list, as shown in Figure 1-6. The new Sample level is in the list.

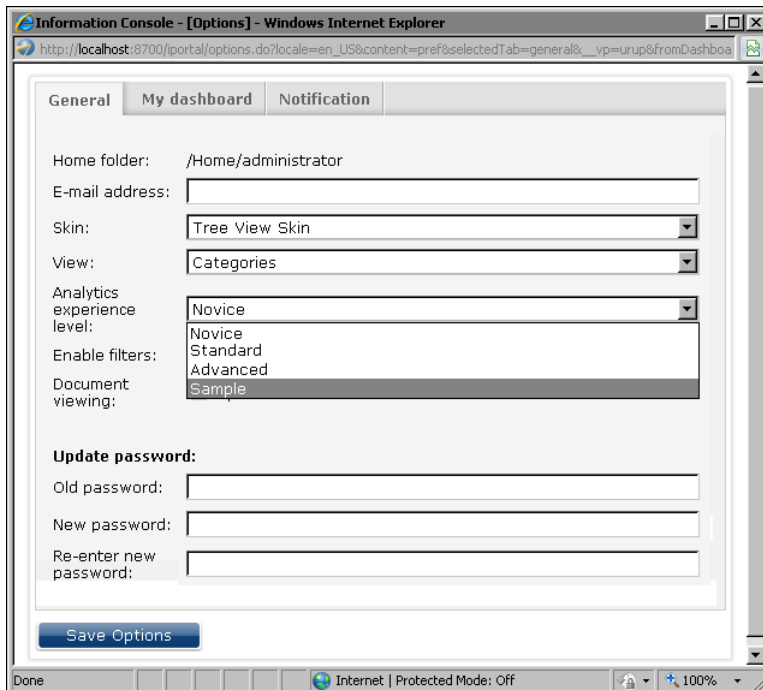


Figure 1-6 Customized Analytics experience levels

Creating a custom Information Console web application

This chapter contains the following topics:

- Information Console web application structure and contents
- Configuring a custom Information Console web application
- Customizing an Information Console web application
- Modifying global style elements

Information Console web application structure and contents

Information Console generates web pages using a set of default JSPs. Actuate Information Console JSPs use cascading style sheets, JavaScript, and custom tags to generate dynamic web page content. The JavaScript and tags provide access to other JSPs, JavaBeans, and Java classes.

The Information Console web application organizes these interoperating components into a Model-View-Controller (MVC) architecture. To operate a web application, the MVC components perform the following functions:

- Model contains the logic for sending requests to and processing responses from the repository. This component is the data model for Information Console.
- View contains the pages that display data prepared by actions. This component is the presentation portion of Information Console.
- Controller contains the servlets that implement actions. This component is the program control logic for Information Console and manages actions initiated from the browser.

The controller maps actions, designated by URLs with the .do extension, to an `actionServlet`. The `actionServlet` is configured with action paths specified in `<Actuate home> \iPortal\iportal\WEB-INF\struts-config.xml`.

Typically, an action path leads to a JSP with parameters as a web resource. Actuate Information Console file and directory names are case sensitive. The first time you use a JSP, your web server compiles it into a servlet. Servlets are compiled Java programs or JSPs that run as part of a network service such as a web server. After compiling a JSP into a servlet, a web server can fulfill subsequent requests quickly, provided that the JSP source does not change between requests.

Users make requests to view the contents of a repository, run and view reports, and so on. Each JSP processes any URL parameters by passing them to JSP tags, including Actuate custom tags or your own custom tags.

You specify the user's Actuate BIRTiServer System and Encyclopedia volume as URL parameters. To specify the locale and time zone to which to connect, use parameter values in an Actuate Information Console request within a URL or by specifying the desired values in the login form. For example, the following URL specifies the `en_US` locale for U.S. English, and the Pacific standard time for the `timezone` parameter:

```
http://localhost:8900/iportal/login.do?locale=en_US&timezone=PST
```

Understanding Information Console directory structure

The Java Server Pages (JSPs) that implement Actuate Information Console URIs are grouped by method into directories under the context root. The context root is the home directory in which an Actuate Information Console web application resides. The default context root for the embedded Information Console for iServer on Windows systems is <Actuate home>\iServer\servletcontainer\iportal and on UNIX and Linux systems is <Actuate home>/iServer/servletcontainer/iportal. The default context root for a separate Information Console installation on Windows systems is <Actuate home>\iPortal\iportal and on UNIX and Linux systems is <Actuate home>/iPortal/iportal. The Information Console context root name in the web or application server's configuration file is iportal. Figure 2-1 shows the Information Console directory structure.

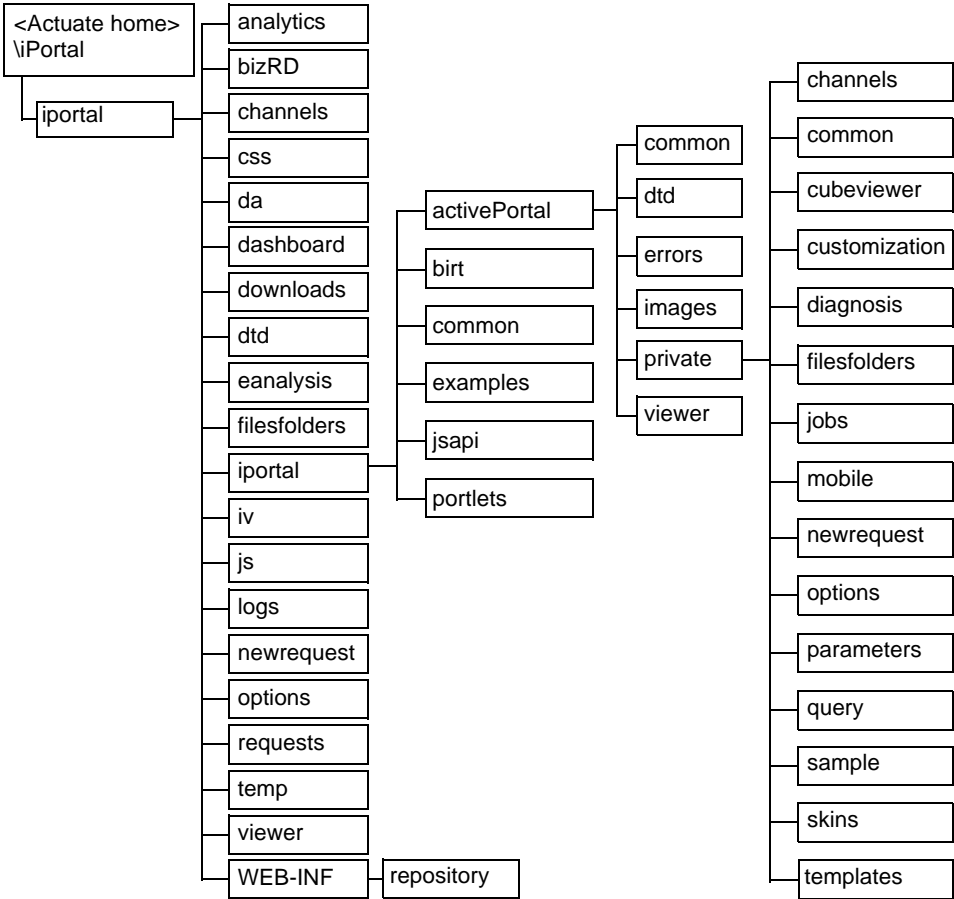


Figure 2-1 Actuate Information Console directory structure

Actuate Information Console URIs convey user requests to Actuate BIRT iServer.

Pages supporting folder navigation and document viewing reside in the <context root>\iportal\activePortal directory. In this directory, pages supporting report viewing reside in the viewer directory, pages serving as templates for other pages reside in the templates directory, and so on. Some directory names exist in the iportal directory and also in the <context root>\iportal\activeportal\private subdirectory. Customize the JSPs in the private subdirectory. The directory of the same name in the iportal directory exists only for backward compatibility. Table 2-1 lists and describes the general iServer\servletcontainer\iportal or iPortal\iportal directories.

Table 2-1 <Context root> directories

Directory	Contents
This directory	landing.jsp, the default page for accessing all Information Console functionality.
analytics	Actuate Analytics support files.
birtAdapter	BIRT Viewer integration files.
bizRD	Pages that support BIRT Studio.
channels	Pages that support channels.
css	Actuate Information Console cascading style sheet (.css) files.
da	BIRT data analyzer support files.
dashboard	Dashboard support files
downloads	Downloaded files.
dtd	Document type definitions.
eanalysis	Actuate e.Analysis Option support files.
filesfolders	Pages that support working with files and folders.
help	Online help.
images	Information Console user interface images and icons.
iportal	The Information Console application.
iv	Pages that support Interactive Viewer.
js	JavaScript files that control specific web page elements such as search, toolbar, and table of contents.
logs	Administrative and SOAP fault log files.
newrequest	Pages that support new requests, such as parameter processing, scheduling, and job status pages.
options	Options-specific pages, such as channels, notification, and options update pages.

Table 2-1 <Context root> directories

Directory	Contents
requests	Pages in this directory provide backward compatibility for custom web applications referencing these pages by URL. Use the action paths and the private\jobs directory for new customization projects.
temp	Working directory for transient content.
viewer	Pages that support report viewing.
WEB-INF	Files that manage session information such as current user login, roles, and volume.

Table 2-2 lists and describes the iportal directories.

Table 2-2 <Context root>/iportal directories

Directory	Contents
activePortal	Pages that support login and authentication and directories for the remaining pages for folder navigation and document usage.
birt	Libraries that support BIRT reports, BIRT Studio, and Interactive Viewer and pages that support BIRT reports.
common	Common elements included in all reporting web pages, such as banner and side menu elements.
jsapi	The Java Report Engine Manager.

Table 2-3 lists and describes the <context root>\iportal\activePortal directories.

Table 2-3 <Context root>/iportal/activePortal directories

Directory	Contents
This directory	Pages that support login and authentication and directories for the remaining folder and document pages for the Information Console application.
common	Common elements included in all reporting web pages, such as banner and side menu elements.
dtd	Document type definitions.
errors	Error pages.
images	Images for reporting web pages, such as buttons, icons, lines, and arrows.

(continues)

Table 2-3 <Context root>/portal/activePortal directories (continued)

Directory	Contents
private	Most Information Console folders and documents web pages. Users cannot directly access pages in this directory using URLs. These pages are customizable.
private \channels	Pages that support channels.
private \common	Common elements included in all reporting web pages, such as banner and side menu elements.
private \cubeviewer	Pages that support viewing Actuate Analytics Option cubes.
private \customization	Pages that support customization of skins.
private \diagnosis	Self-diagnostic utility page.
private \filesfolders	Pages that support working with files and folders.
private\jobs	Pages that support requests such as completed requests, successful submission, and details pages by redirecting.
private\mobile	Pages that support BIRT Mobile subscriptions.
private \newrequest	Pages that support new requests, such as parameter processing, scheduling, and job status pages.
private\options	Options-specific pages, such as channels, notification, and options update pages.
private \parameters	Pages that support table parameters.
private\query	Pages that support Actuate Query functionality.
private\sample	Example custom requester page.
private\skins	Skins definitions.
private \templates	Jakarta Struts template pages that simplify customization by handling common web page structure and functionality for many pages.
viewer	Pages that support report viewing.

Actuate recommends that you group Information Console applications in the home directory of an Actuate distribution to make them easier to locate. Place the context root in whatever location your application requires. To ensure that the JSP engine locates your Information Console application's context root, add its location to your JSP engine's configuration file as a context root path.

Building a custom Information Console context root

Application servers route requests from the user's browser to the configured Information Console web content in a context root. A JSP engine specifies the path for the Information Console context root in a platform-specific configuration file. For example, the Tomcat engine specifies context roots in the `/etc/tomcat/server.xml` file on a UNIX or Linux system and `C:\Program Files (x86)\Apache Software Foundation\Tomcat 6.0\conf\server.xml` file on a Windows system. Other application servers and servlet engines use an analogous file.

You can configure multiple Actuate Information Console context roots on a single server. Each context root can contain a web reporting application that uses a different design. For example, you can create different web reporting applications for particular language groups or departments. The following example is the definition for the default Actuate Information Console context root, `iportal`, from a Tomcat `server.xml` file on a Windows system:

```
<Context
  path="/iportal"
  docBase="C:\Program Files (x86)\Actuate11\iPortal\iportal"
  debug="0"/>
```

The following example is the definition for the default Actuate Information Console context root, `iportal`, from a Tomcat `server.xml` file on a UNIX-based system:

```
<Context
  path="/iportal"
  docBase="/home/user/iPortal/iportal"
  debug="0"/>
```

Actuate Information Console's embedded servlet engine uses an automatic mechanism to discover new web applications. This server provides a quick and convenient environment in which to test your custom Information Console application before deploying to your main application server. To test a custom Information Console application on the embedded servlet engine, you create the context root directory structure in `<Actuate home>iPortal`, then restart the Actuate 11 Apache Tomcat for Information Console service and clear your browser cache.

How to create a new context root

The following example creates a custom web application for MyCorp's Marketing Communications group. The Marketing Communications users use the following URI prefix to access their custom application:

```
http://MyCorp:8700/marcom
```

For example, to access the application's login page, they choose a web page hyperlink with the following URI:

```
http://MyCorp:8700/marcom/login.do
```

- 1 Install Information Console separately. Information Console installed separately is portable but the Information Console embedded in BIRT iServer is not.
- 2 Make a copy of the Actuate Information Console directory structure and give the copy a name related to the context root name.

For example, for the default installation on a Windows machine, copy the directory C:\Program Files\Actuate11\iPortal\iportal, paste it into C:\Program Files\Actuate11\iPortal and rename it marcom.

For the default installation on a UNIX-based machine, copy the directory \$HOME/iPortal/iportal, paste it into \$HOME/iPortal/ and rename it marcom.

- 3 If you are using a server other than Information Console's embedded servlet engine, add your definition to the JSP engine's configuration file. For example, for Tomcat installed on a Windows machine, add the context root, marcom, to the <Information Console Directory>\conf\server.xml file as follows:

```
<Context
  path="/marcom"
  docBase="C:\Program Files\Actuate11\iPortal\marcom"
  debug="0"/>
```

For Tomcat installed on a UNIX-based machine, add the context root, marcom, to the <Information Console Directory>/conf/server.xml file as follows:

```
<Context
  path="/marcom"
  docBase="/home/user/iPortal/marcom"
  debug="0"/>
```

- 4 Restart the application server or JSP engine. For example, to restart Information Console's embedded servlet engine on a Windows XP system, perform the following steps:
 - 1 From the Windows Start menu, choose All Programs→Administrative Tools→Services.
 - 2 On Services, select Actuate 11 Apache Tomcat for Information Console service.
 - 3 From the menu, choose Action→Restart.
 - 4 Close Services.

To restart Information Console's embedded servlet engine on a UNIX-based system, perform the following steps:

- 1 Open a console window.

2 Type the following commands:

```
sh $HOME/iPortal/iportal/actuate_http_service/bin/shutdown.sh  
sh $HOME/iPortal/iportal/actuate_http_service/bin/startup.sh
```

After you stop and restart the server, the Marketing Communications users can access the Actuate Information Console web application called marcom. The application looks like the default Actuate Information Console application because you have not customized its appearance.

Activating a new or custom web application

To activate the changes you make in the Information Console configuration files, content pages, or by creating a new context root, you must restart the web server that runs Information Console and clear the browser cache. For the default Information Console installation, you restart the Actuate 11 Apache Tomcat for Information Console service.

How to restart the Actuate 11 Apache Tomcat for Information Console service on a Windows XP system

- 1 From the Windows Start menu, choose All Programs→Administrative Tools→Services.
- 2 On Services, select Actuate 11 Apache Tomcat for Information Console service.
- 3 From the menu, choose Action→Restart.
- 4 Close Services.

How to clear the browser cache for Microsoft Internet Explorer 7

- 1 Open Microsoft Internet Explorer.
- 2 Choose Tools→Delete Browsing History
- 3 On Delete Browsing History, choose Delete All. Then choose Yes.
- 4 Close Microsoft Internet Explorer.

Configuring a custom Information Console web application

Information Console's configuration determines many of its essential methods. Configuring your web application customizes how it operates internally, and affects the user's experience.

Customize specific pages and operations using the Actuate Information Console web pages, as described in "Customizing an Information Console web application," later in this chapter.

Perform cosmetic customization tasks using the Actuate Information Console skins and style sheets, as described in “Modifying global style elements,” later in this chapter.

Customizing Information Console configuration

Set configuration parameters for the Information Console application to tune performance and to control service and application execution. For example, you can perform the following tasks using configuration parameters:

- Setting the default locale
- Controlling the Message Distribution service load balancing
- Specifying the default Encyclopedia volume and server

You configure the Information Console application by changing configuration file contents, such as `web.xml`. To understand the common configuration files and how each of their entries affect Information Console, see Chapter 3, “Actuate Information Console configuration.”

The following section describes the customization procedure using the text editor.

How to customize Information Console configuration parameters

Use the following procedure to customize configuration parameters for Information Console. In this procedure, it is assumed that `<context root>\WEB-INF\web.xml` is the configuration file.

- 1 Make a backup copy of `web.xml`.
- 2 Using a text editor that supports UTF-8 encoding, edit `web.xml` to change parameter values. Parameter definitions use the following format:

```
<param-name><keyword></param-name>
<param-value><value></param-value>
```

- `<keyword>` is the name of the parameter.
- `<value>` is the parameter value.

Do not enclose the keyword and value within quotes, and use no spaces between `<param-name>`, the keyword or value, and `</param-name>`. For example, the definition for the default locale parameter is:

```
<param-name>DEFAULT_LOCALE</param-name>
<param-value>en_US</param-value>
```

- 3 Save `web.xml`.
- 4 Restart the application server or servlet engine that runs Information Console and clear your browser cache.

Setting the default locale

The default locale and time zone for Information Console are set when you install it. To change the default settings, you modify the values of the `DEFAULT_LOCALE` and `DEFAULT_TIMEZONE` configuration parameters.

How to set a default Information Console locale and time zone

- 1 Using a UTF-8 compliant code editor, open the `web.xml` configuration file.
- 2 Navigate to the lines that define `DEFAULT_LOCALE`, similar to the following code:

```
<param-name>DEFAULT_LOCALE</param-name>
<param-value>en_US</param-value>
```


Change the current locale id, `en_US` in the above example, to the desired locale id in `param-value`. Valid locale id strings are listed in `<context root>\WEB-INF\localemap.xml`.

- 3 Navigate to the lines that define `DEFAULT_TIMEZONE`, similar to the following code:

```
<param-name>DEFAULT_TIMEZONE</param-name>
<param-value>America/Los_Angeles</param-value>
```

Change the current time zone id, `Pacific Standard Time` in the above example, to the desired default time-zone in `param-value`. Valid time zone id strings are listed in `<context root>\WEB-INF\TimeZones.xml`.

- 4 Save `web.xml`.
- 5 Restart the application server or servlet engine that runs Information Console and clear your browser cache.
- 6 Open the Information Console web application. The login page for the custom application appears. A login page with default locale set to `en_GB`, and the default time zone set to `Europe/London`, appears as shown in Figure 2-2.



Language:	English (United Kingdom)
Time zone:	Europe/London

Figure 2-2 The login page for the custom application

Controlling the Message Distribution service load balancing

The default load balancing for Information Console are set to when you install it. To change the default settings, you modify the values of the `MDS_ENABLED` and `MDS_REFRESH_FREQUENCY_SECONDS` configuration parameters.

If you are using third-party load balancing, you will need to refer to their documentation to configure load balancing. See “Understanding Actuate

Information Console load balancing,” in Chapter 1, “Introducing Actuate Information Console.”

How to enable the Message Distribution service

The Message Distribution service (MDS) is enabled by default. This procedure assumes it has been disabled.

- 1 Using a UTF-8 compliant code editor, open the web.xml configuration file.
- 2 Navigate to the lines that define MDS_ENABLED, similar to the following code:

```
<param-name>MDS_ENABLED</param-name>
<param-value>>false</param-value>
```

Change the current value, if it is false, to true.

- 3 Navigate to the lines that define MDS_REFRESH_FREQUENCY_SECONDS, similar to the following code:

```
<param-name>MDS_REFRESH_FREQUENCY_SECONDS</param-name>
<param-value>0</param-value>
```

Change the current refresh frequency in seconds, 0 in the above example, to the desired number of seconds so that MDS will attempt to discover new nodes added to the cluster or remove nodes dropped from the cluster.

- 4 Save web.xml.
- 5 Restart the application server or servlet engine that runs Information Console and clear your browser cache.

Specifying the default Encyclopedia volume and server

The default Encyclopedia volume and server is set when you install Information Console to the local web service and machine name. To use a different Encyclopedia volume and server by default or hide this information in the URL using a volume profile name, you add a profile to the VolumeProfiles.xml configuration file.

How to specify the default Encyclopedia volume and server

- 1 Using a UTF-8 compliant code editor such as JCreator, open the VolumeProfile.xml configuration file
- 2 Navigate to the lines that define the default Profile, similar to the following code:

```
<Profile>
  <Default>true</Default>
  <ProfileName>LocalMachine</ProfileName>
  <RepositoryType>enterprise</RepositoryType>
```

```

    <ServerUrl>http://LocalMachine:8000</ServerUrl>
    <Volume>LocalMachine</Volume>
  </Profile>

```

Navigate to the line that defines Default, and change the value from true to false.

- 3 Create a copy of the entire LocalMachine profile immediately below the LocalMachine profile's </Profile> tag and before the </VolumeProfiles> tag.
- 4 Change the values of your copied profile to the new default Encyclopedia volume and server, similar to the following code:

```

<Profile>
  <Default>true</Default>
  <ProfileName>NewServer</ProfileName>
  <RepositoryType>enterprise</RepositoryType>
  <ServerUrl>http://NewServer:8000</ServerUrl>
  <Volume>NewServer</Volume>
  <DashboardTemplatePath></DashboardTemplatePath>
</Profile>

```

- The value of Default is true, indicating that the profile is the default server profile. Set only one profile Default to true in VolumeProfile.xml, the others must be set to false.
 - The value of ProfileName is a unique name for the server profile.
 - The value of ServerUrl is the URL for the new iServer service to contact by default.
 - The value of Volume is the name of the Encyclopedia volume to access by default.
 - The value of DashboardTemplatePath is an optional repository path for a dashboard file that Information Console loads by default when creating new dashboards.
- 5 Save VolumeProfile.xml. Close the code editor.
 - 6 Restart the application server or servlet engine that runs Information Console and clear your browser cache.
 - 7 Open the Information Console web application. The login page for the custom application appears. The URL will contain the default volume profile information in the VolumeProfile parameter, similar to the following:

```

http://localhost:8900/iportal/login.jsp?
  &__vp=NewServer
  &targetPage=/iportal/getfolderitems.do

```

Modifying text and messages

Actuate Information Console provides text and messages and also passes Actuate BIRT iServer messages to the user. You can customize both Actuate BIRT iServer and Actuate Information Console messages and text. Actuate has created the Actuate Information Console software and resource files in multiple languages. If you need to change the text and messages to translate your Actuate Information Console web application to another language, contact Actuate Corporation.

Customizing Information Console text and messages

Actuate Information Console uses text and messages to communicate with the user. Customize the text of a label to prompt your user with the phrasing that your application needs by changing configuration files in one or more of the files in resources.jar, located in <context root>\WEB-INF\lib\. For example, the default title of the landing page displayed in the title bar and tab text of your web browser is Actuate Information Console, as shown in Figure 2-3.

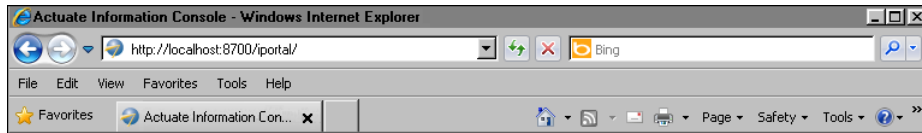


Figure 2-3 Default title bar text of the Information Console landing page

To change this title, change the value of the TITLE_LANDING_PAGE parameter in com\actuate\portal\common\bundle\messages.properties file compressed in the <context root>\WEB-INF\lib\com.actuate.resources.jar. By editing TITLE_LANDING_PAGE, you can customize the marcom web site by replacing the default title with Marcom Information Console, as shown in Figure 2-4.

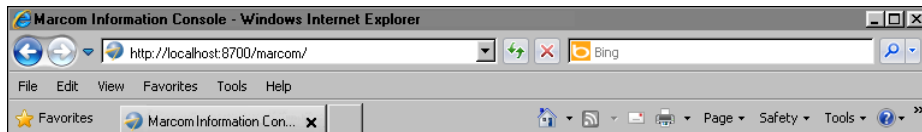


Figure 2-4 Custom title bar text of the Information Console landing page

You can find the method of a particular line of text in the Information Console web application by searching for the relevant message key in the JSPs and examining the related code. To customize a message in other parts of Information Console, you edit the appropriate properties file compressed in resources.jar. Table 2-4 lists the properties files that provide messages and text to particular Information Console page categories.

Information Console inserts additional text using variables. When customizing messages and text, keep the original variables in your text or message, if possible. Variables appear in text and messages in the form {n} where n is a whole number, beginning with 0.

For example, if a user mlee tries to subscribe to a channel but has no available channels other than the user's personal channel, Information Console displays the MSGT_NO_CHANNELS message and its variable from com\actuate\activeportal\resources\ActivePortalResources.properties:

There are no channels available for subscription by {0}.

in the following form:

There are no channels available for subscription by mlee.

How to customize Actuate Information Console text and messages on a Windows system

Use the location of your Information Console installation if it differs from the location used in this example.

- 1 Extract the contents of <context root>\WEB-INF\lib\resources.jar into a temporary directory.

- 1 Open a command window.

- 2 Back up your resources file:

```
cd "C:\Program Files\Actuate11\iPortal\iportal\WEB-INF\lib"
copy com.actuate.resources.jar
      com.actuate.resources.jar.original
```

- 3 Extract the resource file's contents:

```
mkdir C:\ap
cd C:\ap
jar -xf "C:\Program Files\Actuate11\iPortal
      \iportal\WEB-INF\lib\com.actuate.resources.jar"
```

- 4 Leave the command window open.

- 2 Navigate to com\actuate\activeportal\resources and make a backup copy of ActivePortalResources.properties:

```
cd com\actuate\activeportal\resources
copy ActivePortalResources.properties
      ActivePortalResourcesOrig.properties
```

- 3 In a text editor that supports UTF-8 encoding, edit C:\ap\com\actuate\reportcast\resources\ActivePortalResources.properties to add your custom error messages in the following format:

```
<Errorcode>=Example of a message with no variables.
<Errorcode>=Example of a message with a variable {0}.
<Errorcode>=Message with three variables {0}, {1} and {2}.
```

where <Errorcode> is the Actuate error number or constant of the message being customized.

- 4 Save and close the file.

- 5 Rebuild the resources.jar file with your customized ActivePortalResources.properties file:

```
cd C:\ap
jar -cf com.actuate.resources.jar *
move resources.jar "C:\Program Files\Actuate11\iPortal
\iportal\WEB-INF\lib\com.actuate.resources.jar"
```

How to customize Actuate Information Console text and messages on a UNIX or Linux system

Use the location of your Information Console installation if it differs from the location used in this example.

- 1 Extract the contents of resources.jar into a temporary directory:

- 1 Back up your resources file:

```
cd /usr/local/Actuate11/iPortal/iportal/WEB-INF/lib
cp com.actuate.resources.jar
com.actuate.resources.jar.original
```

- 2 Extract the resource file's contents:

```
mkdir ap
cd ap
jar -xf /usr/local/Actuate11/iPortal/iportal/WEB-INF/lib
/com.actuate.resources.jar
```

- 2 Navigate to com/actuate/activeportal/resources and make a backup copy of ActivePortalResources.properties:

```
cd com/actuate/activeportal/resources
cp ActivePortalResources.properties
ActivePortalResourcesOrig.properties
```

- 3 In a text editor that supports UTF-8 encoding, edit ap/com/actuate/reportcast/resources/ActivePortalResources.properties to add your custom error messages in the following format:

```
<Errorcode>=Example of a message with no variables.
<Errorcode>=Example of a message with a variable {0}.
<Errorcode>=Message with three variables {0}, {1} and {2}.
```

where <Errorcode> is the Actuate error number or constant of the message being customized.

- 4 Save and close the file.
- 5 Rebuild the resources.jar file with your customized ActivePortalResources.properties file:

```
jar -cf resources.jar *
mv resources.jar /usr/local/Actuate11/iPortal/iportal
/WEB-INF/lib/com.actuate.resources.jar
```


Customizing Actuate BIRT iServer error messages

Actuate Information Console uses SOAP messages to communicate with the Actuate BIRT iServer. You can customize the message text of an Actuate BIRT iServer error message before Information Console displays it to the user. For example, the following URL attempts to schedule a job for a report that is not in the repository:

```
http://localhost:8700/iportal/submitjob.do?requesttype=scheduled&
executableName=BadFileName.x
```

Information Console retrieves an iServer error message, as shown in Figure 2-5.

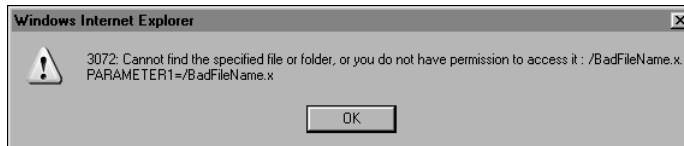


Figure 2-5 iServer error message for a missing file.

To customize a message, you edit `ErrorMessages.properties`, following the procedures described later in this section. This file contains customized error messages. For a full list of all BIRT iServer error messages, see `<context root>\WEB-INF\ErrorMessages.txt`. This file contains the error code, error level, and the English text of every message. When you customize `ErrorMessages.properties`, use the error code for the message from `ErrorMessages.txt`.

Information Console inserts context-specific text to an error message using variables. When changing message text, maintain the original variables in your new message, if possible. For the best results, follow the format of the original message exactly to maintain the number and order of the variables. Variables appear in message text as `{n}` where `n` is a whole number, beginning with 0.

For example, the URL for a missing file produces error 3072, and you can change the entry for error 3072 to something similar to the following:

```
3072 = {0} is a bad file name or the file does not exist.
```

Using the erroneous URL above with this custom message results in a new message, as shown in Figure 2-6.



Figure 2-6 Custom iServer error message for a missing file.

How to customize Actuate BIRT iServer error messages on a Windows system

Use the location of your own Actuate Information Console installation if it differs from the location used in this example.

- 1 Extract the contents of <context root>\WEB-INF\lib\resources.jar into a temporary directory.

- 1 Open a command window.

- 2 Back up your resources file:

```
cd "C:\Program Files\Actuate11\iPortal\iportal\WEB-INF\lib"
copy com.actuate.resources.jar
      com.actuate.resources.jar.original
```

- 3 Extract the resource file's contents:

```
mkdir C:\ap
cd C:\ap
jar -xf "C:\Program Files\Actuate11\iPortal
      \iportal\WEB-INF\lib\com.actuate.resources.jar"
```

- 4 Leave the command window open.

- 2 Navigate to com\actuate\reportcast\resources and make a backup copy of ErrorMessage.properties:

```
cd com\actuate\reportcast\resources
copy ErrorMessage.properties ErrorMessageOrig.properties
```

- 3 In a text editor that supports UTF-8 encoding, edit C:\ap\com\actuate\reportcast\resources\ErrorMessage.properties to add your custom error messages in the following format:

```
<Errorcode>=Example of a message with no variables.
<Errorcode>=Example of a message with a variable {0}.
<Errorcode>=Message with three variables {0}, {1} and {2}.
```

where <Errorcode> is the Actuate error number or constant of the message being customized.

- 4 Save and close the file.

- 5 Rebuild the resources.jar file with your customized ErrorMessage.properties file:

```
jar -cf resources.jar *
move resources.jar "C:\Program Files\Actuate11\iPortal
      \iportal\WEB-INF\lib\com.actuate.resources.jar"
```

How to customize Actuate BIRT iServer error messages on a UNIX or Linux system

Use the location of your Information Console installation if it differs from the location used in this example.

- 1 Extract the contents of resources.jar into a temporary directory:
 - 1 Back up your resources file:

```
cd /usr/local/Actuate11/iPortal/iportal/WEB-INF/lib
cp com.actuate.resources.jar
   com.actuate.resources.jar.original
```
 - 2 Extract the resource file's contents:

```
mkdir ap
cd ap
jar -xf /usr/local/Actuate11/iPortal/iportal/WEB-INF/lib
    /com.actuate.resources.jar
```
- 2 Navigate to com/actuate/activeportal/resources and make a backup copy of ErrorMessage.properties:

```
cd com/actuate/activeportal/resources
cp ErrorMessage.properties ErrorMessageOrig.properties
```
- 3 In a text editor that supports UTF-8 encoding, edit ap/com/actuate/reportcast/resources/ErrorMessage.properties to add your custom error messages in the following format:

```
<Errorcode>=Example of a message with no variables.
<Errorcode>=Example of a message with a variable {0}.
<Errorcode>=Message with three variables {0}, {1} and {2}.
```

where <Errorcode> is the Actuate error number or constant of the message being customized.
- 4 Save and close the file.
- 5 Rebuild the resources.jar file with your customized ErrorMessage.properties file:

```
jar -cf resources.jar *
mv resources.jar /usr/local/Actuate11/iPortal/iportal/WEB-INF
    /lib/com.actuate.resources.jar
```

Customizing an Information Console web application

To perform most cosmetic customization tasks, use the Actuate Information Console skin manager. The skin manager supports using skins to change typically customized images, colors, and fonts in Actuate Information Console web pages. You also can customize aspects of Information Console that are not supported by the skin manager by modifying the Information Console files manually.

Actuate Information Console supports customization of the landing page, `<context root>\landing.jsp`, and the appearance of the pages in My Documents, BIRT Studio, and the interactive viewer for BIRT reports.

You use knowledge of the following standard languages and frameworks to customize an Information Console web application manually:

- **Cascading style sheet (.css) files**
CSS files define fonts, colors, and other visual design attributes of an Information Console web application. For information about modifying style sheets, see “Modifying global style elements,” later in this chapter.
- **Hypertext markup language (HTML)**
HTML handles links and the presentation of text and graphics in web pages. Information Console incorporates HTML code in its JavaServer pages.
- **Jakarta Struts Framework**
Jakarta Struts Framework is an open source framework for building web applications. Based on standard technologies, Struts enables the Information Console Model-View-Controller design. For more information about Struts, access the following URL:

`http://jakarta.apache.org/struts`
- **Java**
Information Console uses Java classes to provide functionality. You can create your own Java classes for your custom web application. For more information on the Information Console Java classes, see Chapter 8, “Actuate Information Console JavaBeans.”
- **JavaScript**
JavaScript is an interpreted, object-oriented language that facilitates embedding executable content in web pages. It provides strong tools for interacting with web browsers.
- **JavaServer Pages**
The JavaServer Pages (JSP) extension of the Java Servlet API facilitates the separation of page design from business logic. JSPs are a platform-independent solution. Information Console web pages are defined primarily by JSPs. For more information about the Actuate JavaServer Pages, see Chapter 4, “Actuate Information Console URIs.”

Actuate recommends that you use the skin manager to customize as much as possible and then handle any remaining customization tasks manually.

Modifying the landing page

To modify the appearance of the landing page, use custom styles as described later in this chapter. The landing page uses the same cascading style sheets files

as the other Actuate Information Console JSPs. Figure 2-7 shows some of the classes that define various elements of the landing page. Where possible, modify these styles by using the customization pages for skins.

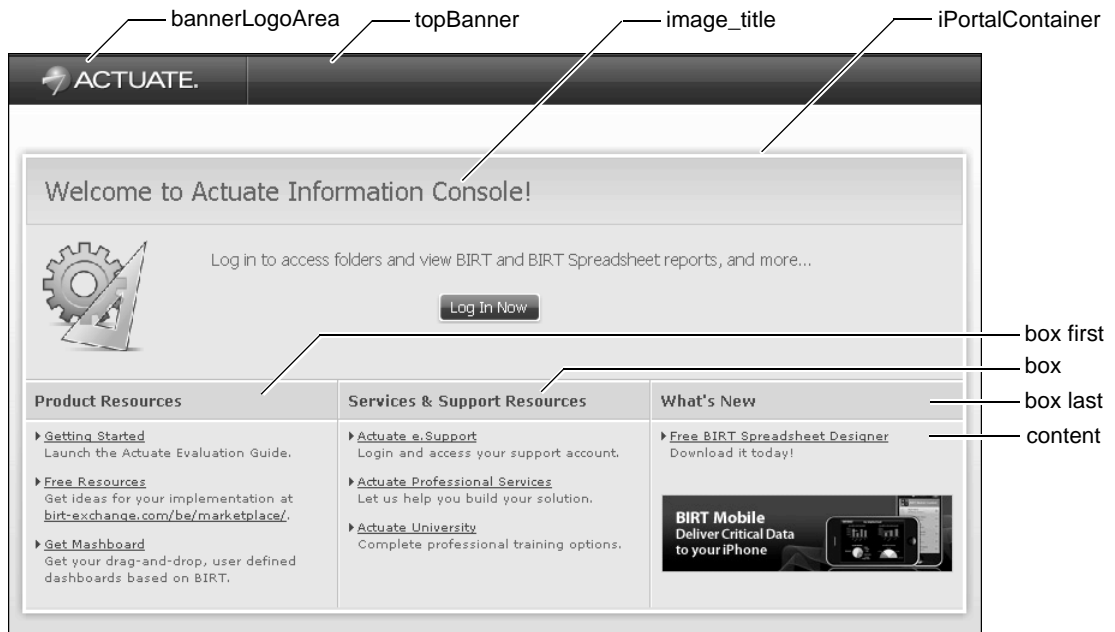


Figure 2-7 Classes used on the default landing page

Use the contents of the `<body>` element in `landing.jsp` to customize the branding images, the welcome text in the banner, and the list of links that appears at the left of the default landing page. The code comments indicate which portion of the page can be modified. For example, the code after the comment `<%-- THE UPPER SECTION --%>` refers to the banner area at the top of the page.

Viewing modifications to a custom web application

After making changes to your Information Console web application, you need to view the changes. Caching in the browser or your application server can interfere with seeing the changes you made. After changing an Information Console application, complete these general tasks in order:

- Save any files involved in the change.
- Refresh the browser page.
- If you do not see changes you made in a JSP or XML file, complete the following tasks in order:
 - Shut down the JSP engine.

- Clear the JSP engine's cache or work directory to ensure that the JSP engine picks up your changes. For example, to force Information Console's embedded servlet engine to use the changed files, delete all files from C:\Program Files\Actuate11\iPortal\work and clear the web browser's cache.
- Restart the JSP engine.
- If you do not see changes you made in a cascading style sheet file or a JavaScript file, clear the web browser's cache, then refresh the page.

Your changes appear in the web browser.

Locating existing pages and linking in new pages

To locate an existing page, navigate to that page and examine the URI in the address field of your browser. If the URI contains a JSP name, go to that JSP file. If the URI contains an action path, search struts-config.xml for that action path without the .do extension, or look up the action path in Chapter 4, "Actuate Information Console URIs."

An action path is a uniform resource identifier (URI) called directly by Information Console or by a user to access the Information Console functionality. <context root>\WEB-INF\struts-config.xml contains the action path specifications.

An action path specifies a JSP to use in response to user controls. An action path uses the results of an Action class to determine the next action path to use or the next JSP to display. Typically, an action class indicates one action path or JSP if the execution succeeds and a different action path or JSP if the execution causes an error. In the following code sample, if the AcGetFolderItemsAction JavaBean executes successfully, the next JSP to display is <context root>\iportal\activePortal\private\filesfolders\filefolderlist.jsp:

```
<!-- Process getfolderitems -->
<action
  attribute="fileListActionForm"
  name="fileListActionForm"
  path="/getfolderitems"
  scope="request"
  type="com.actuate.activeportal.actions.AcGetFolderItemsAction"
  validate="false">
  <forward name="success"
    path="/iportal/activePortal/private/filesfolders
      /filefolderlist.jsp" />
  <forward name="dashboard" path="/dashboard" redirect="true" />
</action>
```

In the preceding example, the path for an error result is not listed. This means that it defaults to the definition in the global forwards section of struts-config.xml as a when an error occurs:

```
<forward name="error" path="/iportal/activePortal/private/common
    /errors/errorpage.jsp"/>
```

To add a forward command that activates when the JavaBean returns another result, such as viewroi, you can include a forward for that result to direct it accordingly, as shown in the following example:

```
<forward name="viewroi"
    path="/iportal/activePortal/viewer/viewframeset.jsp"
    redirect="true" />
```

To add a new web page to Information Console, you change the navigation in struts-config.xml to use the new JSP or path. You can change an existing input page or forward page specification in an action path to your new page, or you can create a new action path that forwards to your page. If you create a new action path, you can change another action path to forward to your new path or you can modify or create links on web pages to specify your new action path. The following action path always navigates to welcome.jsp when another action path, link, or URL invokes it:

```
<!-- Process welcome -->
<action path="/welcome"
    forward="/iportal/activePortal/private/welcome.jsp"
    name="welcome">
</action>\
```

For more information on action paths and Jakarta Struts, go to the following URL:

<http://jakarta.apache.org/struts>

Obtaining information about the user and the session

Typically, new Actuate Information Console web pages need access to session information. Your application server and Information Console store information about the session that you can use in your web pages. Obtain the serverURL, volume, and other information from your application server using the JSP request variable, as shown in the following example.

```
String volume = request.getParameter("volume");
String serverURL = request.getParameter("serverurl");
String userId = request.getParameter("userid");
String roxReport = request.getParameter("report");
```

You can also obtain the context root path from your application server, as shown in the following code:

```
String contextRoot = request.getContextPath();
```

Additionally, Actuate Information Console stores a wide variety of information about the session in `UserInfoBean`. To access `UserInfoBean`, place the following line of code near the top of your JSP:

```
<jsp:useBean id="userinfobean"
  class="com.actuate.activeportal.beans.UserInfoBean"
  scope="session"/>
```

After this line, you can access information in the `JavaBean` by the appropriate get method. The most important method for new pages is the `getIportalid()` method. This method retrieves the user's authentication ID with the server. This ID is based on the server, volume, and user name supplied on the login page.

To write generic code, you need to determine whether your application is running. Information Console includes a utility class, `iPortalRepository`, that provides this information. To access this class in your JSP, place the following code at the head of your JSP:

```
<%@ page
  import="com.actuate.iportal.session.iPortalRepository" %>
```

Then use code similar to the following line to check the repository type:

```
boolean isEnterprise =
  iPortalRepository.REPOSITORY_ENCYCLOPEDIA.equalsIgnoreCase(
    userinfobean.getRepositoryType());
```

Use the authentication ID and the repository type to access the server with JSP custom Actuate tags and calls to Information Console beans, as shown in the following examples:

```
String authenticationID = userinfobean.getIportalid();
String folderPath = userinfobean.getCurrentfolder();
jobDetailURL += StaticFuncs.encode(userinfobean.getUserid());
com.actuate.reportcast.utils.AcLocale acLocale =
  userinfobean.getAcLocale();
TimeZone timeZone = userinfobean.getTimezone();
boolean isEnterprise =
  iPortalRepository.REPOSITORY_ENCYCLOPEDIA.equalsIgnoreCase(
    userinfobean.getRepositoryType());
String serverURL =
  ( isEnterprise | userinfobean.getServerurl() | " " );
String userVolume =
  ( isEnterprise | userinfobean.getVolume() | " " );
```

Customizing accessible files and page structure using templates

Actuate Information Console uses Jakarta Struts templates to simplify JSP code and customization. Information Console templates handle overall page organization, access to Jakarta Struts custom tag libraries, and access to common

CSS and JavaScript files. The login page and landing page do not use a template. Table 2-4 describes the Information Console templates.

Table 2-4 Actuate Information Console Struts templates

Template	Method
dashboardtemplate.jsp	Used for BIRT 360 dashboards.
simpletemplate.jsp	Used for errors, confirmations, and other simple pages
querytemplate.jsp	Used by most Actuate Query pages
template.jsp	Used by all other pages except the login and landing pages

Each Actuate Information Console skin has its own version of these templates, besides the dashboard template, in <context root>\iportal\activePortal\private\skins\<skin name>\templates. The set of templates in <context root>\iportal\activePortal\templates sets up several JavaBeans and then accesses the template of the same name for the user's selected skin. The dashboard template is located in <context root>\dashboard\jsp, along with the dashboard JSP files.

Specifying a template and template elements

To use a template and template elements, a page uses the Jakarta Struts custom template tags, described in Table 2-5.

Table 2-5 Struts template tags

Template tag	Method
template:insert	Specifies the template to use
template:put	Specifies the text or file to use for a template element such as the name, banner, side menu, or content elements

The custom template tags define the JSPs to use for the template and the custom elements that the template specifies to build the user interface. For example, the template:insert tag in the following code applies querytemplate.jsp settings to the page. The first template:put tag accesses the localized string for the title of the page. The remaining template:put tags specify that the template use banner and content elements using the files specified in each tag.

The following code example is an extract from <context root>\iportal\activePortal\private\newrequest\newrequest.jsp.

```
<template:insert template="/iportal/activePortal/private
  /templates/querytemplate.jsp">
```

```

<template:put name="title" direct="true">
  <bean:message bundle="iportalResources"
    key="<%=whichTitle%>" arg0="<%=pageTitle%>" />
</template:put>
<template:put name="banner"
  content="/iportal/activePortal/private/common/banner.jsp" />
<template:put name="content"
  content="/iportal/activePortal/private/newrequest
/newrequestpage.jsp" />
</template:insert>

```

The following tables show JSPs affected by template changes. Table 2-6 lists the Information Console templates and the pages that use them.

Table 2-6 Templates for JSPs

Template	JSPs in iportal\activePortal\private
querytemplate.jsp	jobs\getrequesterjobdetails.jsp jobs\requesterjoboperationstatus.jsp newrequest\newrequest.jsp newrequest\submitjobstatus.jsp query\create.jsp query\execute.jsp
simpletemplate.jsp	common\errors\errorpage.jsp customization\fileupload.jsp newrequest\newrequest2.jsp
simpletemplate.jsp	query\confirmation.jsp query\fileexists.jsp query\runconfirmation.jsp
template.jsp	channels\channellist.jsp channels\channelnoticelist.jsp channels\channeloperationstatus.jsp channels\channelsubscribe.jsp customization\skinedit.jsp customization\skinmanager.jsp filesfolders\createfolder.jsp filesfolders\deletefilestatus.jsp filesfolders\filedetail.jsp filesfolders\filefolderlist.jsp filesfolders\privilege.jsp filesfolders\search\filefolderlist.jsp jobs\getjobdetails.jsp jobs\joboperationstatus.jsp jobs\selectjobs.jsp newrequest\submitjobstatus.jsp options\options.jsp

About the dashboard template

The JSPs in <context root>\dashboard\jsp define the dashboard interface for the Actuate BIRT 360 option. The dashboard template applies to the banner and content when the dashboard is visible as defined by dashboard.jsp, as follows:

```
<template:insert template="/dashboard/jsp/dashboardtemplate.jsp">
  <template:put name="title" direct="true">
    <bean:message bundle="iportalResources" key="MSGT_BROWSER"
      arg0="<%= pageTitle %>" />
  </template:put>
  <template:put name="banner" content="<%= bannerUrl %>" />
  <template:put name="content" content="/dashboard/jsp/
    dashboardcontent.jsp" />
</template:insert>
```

The contents of a separate page or pages displayed on a dashboard use templates defined by that page's code, but the surrounding interface elements adhere to the dashboard template.

Changing a template

Make changes to all pages that use a particular template by changing only the template. You can add or remove lines in the template that make cascading style sheets, JavaScript files, and other resources accessible to all pages that use the template. Customize the overall structure of all pages that use a template by moving, resizing, or removing the HTML, JSP, and Jakarta Struts code describing the layout of the web pages that use the template.

For example, the innerTable of <context root>\iportal\activePortal\private\skins\treeview\templates\template.jsp specifies various HTML commands and embedded Jakarta Struts tags that populate the content frame. The inner banner with the breadcrumb is in the top row. The second row contains the content page.

```
<table class="innerTable" border="0" cellspacing="0"
  cellpadding="0">
  <% if (!"false".equalsIgnoreCase(showBreadcrumb)) { %>
  <tr>
    <td class="allBreadcrumbs">
      <jsp:include page="<%= breadcrumb %>" flush="true" >
      <jsp:param name="fromDashboard" value="<%= fromDashboard %>" />
      <jsp:param name="showBanner" value="<%= showBanner %>" />
      <jsp:param name="showSideBar" value="<%= showSideBar %>" />
      <jsp:param name="showBreadcrumb" value="<%= showBreadcrumb %>"
      />
    </jsp:include>
    </td>
  </tr>
```

```

<% } %>
<tr>
  <td class="fileFolderListContent" valign="top">
    <div>
      <template:get name="content" flush="true"/>
    </div>
  </td>
</tr>
</table>

```

The breadcrumb, or navigation trail, is a link or set of links. On a document page, the breadcrumb displays the repository and any folders and pages you access. Use any of these items as a link to return to that level. For a jobs or channels page, the breadcrumb supports direct access to a document page.

To implement the expandable tree, a frameset in <context root>\iportal\activePortal\private\skins\treeview\templates\template.jsp specifies the sidebar and content frames using HTML and embedded Jakarta Struts tags that define the content.

```

<FRAMESET cols="20%,80%" border="1"
  onload="if (typeof(bodyOnload) != 'undefined') bodyOnload();">
  <FRAME src="<html:rewrite page="<%= sidebar %>" />"
    name="<%=htmlSideFrameName%>"
    id="<%=htmlSideFrameId%>"
    scrolling="auto"
  />
  <FRAME src="<html:rewrite href="<%= contentURL %>" />"
    name="main" scrolling="auto"
  />
</FRAMESET>

```

Modifying existing content or creating new content

You can modify the content of an existing page or create new pages for linking in to your custom web application. Typically, a web page has one JSP that implements a template and another JSP to implements the content to display according to the template's structure. For example, the following code specifies that the template's content element on a web page uses the JSP code in <context root>\iportal\activePortal\private\newrequest\newrequestpage.jsp:

```

<template:put name="content" content="/iportal/activePortal
/private/newrequest/newrequestpage.jsp" />

```

The content JSP contains the code that creates the page-specific content and functionality. The newrequestpag.jsp contains code that places page-specific text, graphics, links, and other functionality on the page. You can use HTML code, JSP code, JSP built-in tags, Jakarta Struts tags, Actuate servlets, Actuate custom tags, Actuate JavaBeans, CSS, and JavaScript methods to obtain data and present

information on the page. For information about how to use these features, see “Customizing an Information Console web application,” later in this chapter.

The default Actuate Information Console pages use HTML tables to provide formatting for each page. The tables are often nested. Individual files include other files that define elements, such as the <TABLE> declaration. As you modify the pages to suit your needs, verify that the Actuate Information Console pages for tasks, such as logging in, listing folders and files, and viewing and requesting reports appear correctly in your web browser.

When using relative hyperlinks in your HTML code, ensure that any files to which you refer are available to Actuate Information Console. Information Console resolves relative hyperlinks from the context root. For example, in the standard Information Console installation, the following code refers to an images directory at the same level as the Information Console context root directory:

```
<A HREF="../images/myimage.gif">
```

All Actuate Information Console requests require action paths to have certain names. Similarly, the action paths require JSP files to have certain names and to reside in a particular directory under the context root. Do not rename the default files provided with Information Console without making the corresponding change to struts-config.xml. If you do not change the file name consistently in all places, Information Console cannot locate your custom files.

Modifying global style elements

Although JSPs can use HTML to set colors, fonts, and other stylistic elements directly, the JSPs also use cascading style sheets (CSS), templates, and shared images to control the global styles of an Information Console web application. To modify the appearance of the entire Information Console web application, change global style elements.

Global styles can change more than the appearance of Actuate Information Console. For example, to view search results with HKSCS characters in an English locale, change the .searchresultlink style’s font from Arial to MingLiU_HKSCS. This style change only affects the search results.

Customizing Actuate Information Console using skins

Actuate Information Console skins support customizing the Actuate Information Console colors, fonts, and images in the graphical user interface (GUI) for the pages in My Documents, BIRT Studio, and the interactive viewer for BIRT reports. A skin consists of images, cascading style sheets, JavaScript, and template files used to define the GUI. Actuate Information Console installs with three skins. Only users with the Administrator functionality level can customize skins.

Using skins

To select a different Information Console skin, a user chooses Options, then selects a name from the Skin drop-down list, as shown in Figure 2-8.

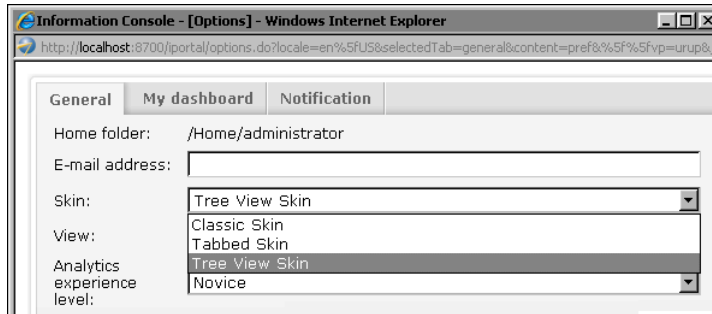


Figure 2-8 Default skins choices

Actuate Information Console provides three default skins:

- Use the Classic skin to view Documents, My Jobs, and Channels as buttons in the side menu, as shown in Figure 2-9.

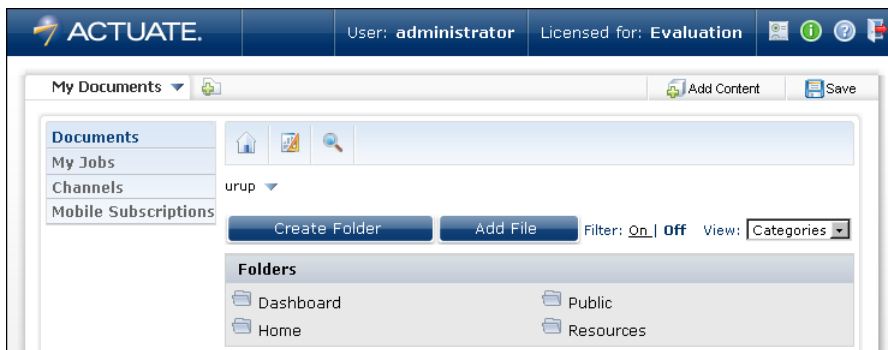


Figure 2-9 Classic skin

- Use the Tabbed skin to view Documents, My Jobs, and Channels as tabs on the banner at the top of the page, as shown in Figure 2-10.
- Use the Treeview skin to view Documents, My Jobs, and Channels in the side menu as a hierarchical view. The folders view starts from the root folder of an Encyclopedia volume. This hierarchical view is similar to that of Windows Explorer, as shown in Figure 2-11, and is the default skin. The Treeview skin does not support placement in an iFrame.

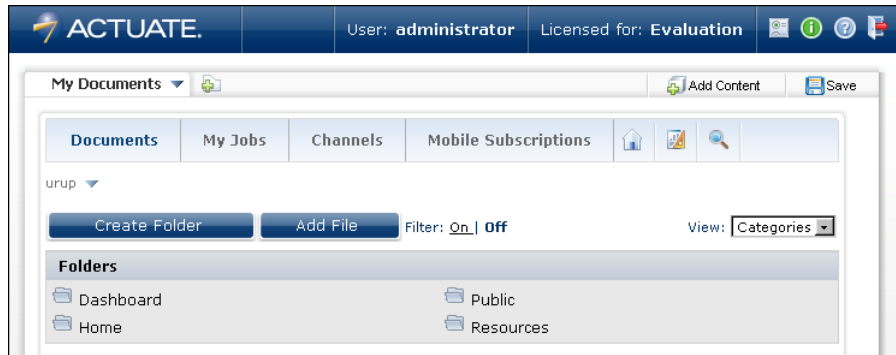


Figure 2-10 Tabbed skin

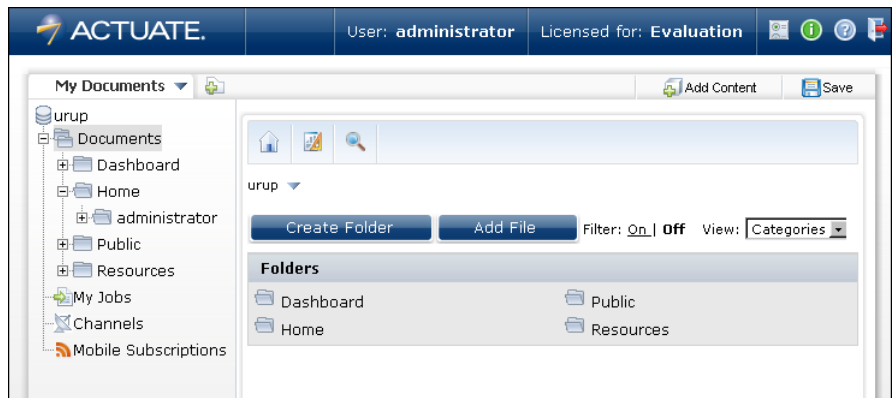


Figure 2-11 Treeview skin

Managing skins using the skin manager

Users with the Administrator functionality level manage skins for all users. The skin manager controls skins and their settings. To access the skin manager, choose Customization on the Information Console banner as shown in Figure 2-12.

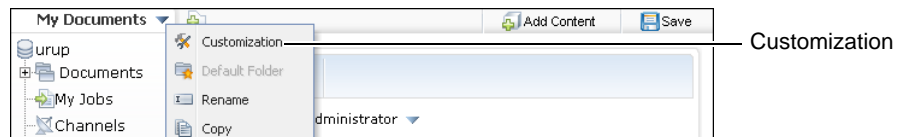


Figure 2-12 Information Console banner, showing Customization menu option
The default skin manager looks like the one in Figure 2-13.


Skins > Manager			
Name	Description	Default	Public
classic	Classic Skin		<input checked="" type="checkbox"/>
tabbed	Tabbed Skin		<input checked="" type="checkbox"/>
treeview	Tree View Skin		<input checked="" type="checkbox"/>

Figure 2-13 skin manager, showing the default skins

Table 2-7 describes the features of the skin manager.

Table 2-7 skin manager functionality

Feature	Description
Clone	Adds a copy of the skin to the table as private.
Customize	Displays the Skin—Customize page to allow customizing for that skin. The skins shipped by default with Actuate Information Console cannot be customized.
Default	Selects the skin used for new users by default without affecting existing users. Setting a skin to Default makes it public and disables its Public check box.
Delete	Deletes the skin after confirmation. Skins shipped with Actuate Information Console and the default skin cannot be deleted. To delete the current default skin, first choose another skin as the default.
Preview	Applies the skin immediately. When the current session times out, the skin reverts back to the user's original skin. The user's current skin is shown in bold text.
Public	Makes the skin available for all users by adding the skin to the list on the Options page. If a public skin becomes private, users using the skin revert to the default skin. The default skin is always public.

Customizing and cloning skins

Actuate Information Console ships with three standard skins. You cannot customize the standard skins. You can customize any skin clone, or copy, you create. The skins that Information Console provides may be modified during an upgrade, but any skin you create is preserved during upgrades. To customize any of the three standard skins, clone the skin to create a copy, and then customize the clone. When cloning a skin, select the skin that is closest to the required appearance. Perform additional customizations to a cloned skin at any time.

Table 2-8 lists the GUI components of cloned skins that you can customize.

Table 2-8 Customizable components of skins

Item	Customizable components
Colors	Banner, footer, side menu, tabbed dialogs, pop-up menus, viewer, templates
Fonts	Multiple font families in order of preference
General	Skin description that appears on the Options page
Images	Banner logo, My Folder icon, volume icon, open and closed folder icons

The skin description appears on the Options page to identify the skin to users.

Colors are grouped into categories according to the GUI area they affect. Table 2-9 lists ways to specify colors.

Table 2-9 Techniques for specifying colors

Specification	Description
Color code	Type a standard HTML color or hexadecimal RGB value.
Red Green Blue	Type individual RGB values, from 0 to 255.
Pick a color	Select from a palette of available colors.

To customize images for a skin, upload GIF or JPEG files to Actuate Information Console to replace the existing images. Images are grouped in categories by their GUI component. The categories and the images that you can replace depend upon the type of skin. For example, more images are available to customize in a skin based on the Treeview than a skin based on the Classic skin. Icon images must be consistent in size with the images they replace. Most icons supplied with Actuate Information Console are either 32x32 or 16x16 pixels.

After making changes to a skin, use the preview functionality to view different Actuate Information Console pages to show the skin's current appearance. By checking multiple pages, you identify the areas that need modification.

How to clone a skin

Use the following procedure to create a new skin, based on an existing skin.

- 1 Log in to the documents web pages as an administrator-level user. Choose Customization.
- 2 In the skin manager, choose Clone on an existing skin, as indicated in Figure 2-14.

Skins > Manager							
Name	Description	Default	Public				
classic	Classic Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
tabbed	Tabbed Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
treeview	Tree View Skin	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview

Figure 2-14 Clone functionality for a skin

- At the prompt, type a name for the new skin. Choose OK. The new skin appears in the list of available skins, as shown in Figure 2-15. Do not select Public or Default until you have finished the skin development.

Skins > Manager							
Name	Description	Default	Public				
classic	Classic Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
tabbed	Tabbed Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
treeview	Tree View Skin	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
Clone of treeview	Clone of treeview	<input type="radio"/>	<input type="checkbox"/>	Customize	Clone	Delete	Preview

Figure 2-15 The skin manager, showing a cloned skin

How to customize a skin

The following procedure customizes the skin created in “How to clone a skin,” earlier in this chapter.

- In the skin manager, on the skin to change, choose Preview, as indicated in Figure 2-16. The appearance of Actuate Information Console pages changes to match the selected skin.

Skins > Manager							
Name	Description	Default	Public				
classic	Classic Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
tabbed	Tabbed Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
treeview	Tree View Skin	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
Clone of treeview	Clone of treeview	<input type="radio"/>	<input type="checkbox"/>	Customize	Clone	Delete	Preview

Figure 2-16 Preview functionality for a skin

- On the skin to change, choose Customize, as indicated in Figure 2-17.

Customize

Skins > Manager							
Name	Description	Default	Public				
classic	Classic Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
tabbed	Tabbed Skin	<input type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
treeview	Tree View Skin	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	Customize	Clone	Delete	Preview
Clone of treeview	Clone of treeview	<input type="radio"/>	<input type="checkbox"/>	Customize	Clone	Delete	Preview

Figure 2-17 Customize functionality for a cloned skin

- 3 On Skins—Customize—General, change the skin description to a unique value that conveys meaning to your users, as shown in Figure 2-18.

Skins > Customize: Clone of treeview

General Images Colors Fonts

Skin Name: Clone of treeview

Description:

Figure 2-18 General pane for skin customization

- 4 Select Images. The Images pane appears, as shown in Figure 2-19. Choose a category name to see the images in that category.

Skins > Customize: Clone of treeview

General Images Colors Fonts

Asterisk (*) items will take effect only if their corresponding items are not set in "Colors" tab.








Name	Preview	
<u>General</u>		
<u>Banner</u>		
Logo		Upload File...
Blue Logo		Upload File...
Top banner background *		Upload File...
Top banner divider background *		Upload File...
Inner banner middle background *		Upload File...
Options		Upload File...
About		Upload File...

Figure 2-19 Images pane for skin customization

- 5 Select Colors. The Colors pane appears, as shown in Figure 2-20. The categories shown depend upon the type of skin. Choose a category name to toggle between showing and hiding the list of colors in that category.

Skins > Customize: Clone of treeview

General Images Colors Fonts

Name	Color Code	Red	Green	Blue	
<u>General</u>					
<input type="checkbox"/> Landing and Login pages background *		0	0	0	Pick Color...
<input type="checkbox"/> Body background	White	255	255	255	Pick Color...
<input checked="" type="checkbox"/> Text Color	black	0	0	0	Pick Color...
<input type="checkbox"/> Panel background color	#DADADA	218	218	218	Pick Color...
<input type="checkbox"/> Listing content background color	#E8E8E8	232	232	232	Pick Color...
<input type="checkbox"/> Listing content highlight color	white	255	255	255	Pick Color...
<input type="checkbox"/> Panel border color	#b5b29c	181	178	156	Pick Color...
<input type="checkbox"/> Breadcrumbs background color	white	255	255	255	Pick Color...

Figure 2-20 Colors pane for skin customization

- 6 Select Fonts. The Fonts pane appears, as shown in Figure 2-21. On Name, select General. Font Family appears. Specify one or more font families to use. Actuate Information Console uses the first font in the list found on the machine where Actuate Information Console is deployed.

Skins > Customize: Clone of treeview

General Images Colors Fonts

Name	Preview	Font family in order of preference
<u>General</u>		
Font Family	Sample Text	Verdana, Arial, Helvetica, sans-serif

Figure 2-21 Fonts pane for skin customization

Choose OK.

- 7 To make the new skin available to all users, on the Skins > Manager page, select Public for your new skin.
- 8 To make the skin the default skin for all users, select Default. Figure 2-22 shows a custom skin, Clone of classic, based on the Classic skin.

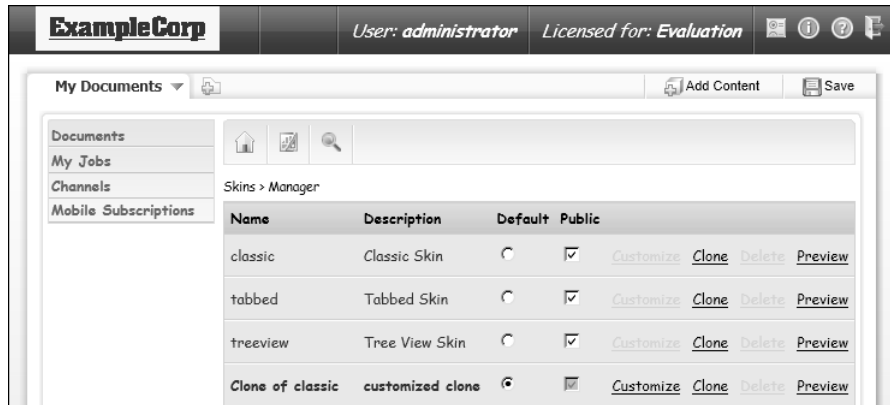


Figure 2-22 An example of a custom skin

Understanding style definition files

Additional style definitions for each provided skin come from `<context root>\iportal\activePortal\private\skins\<skin name>\css\skinstyles.css`. Add more styles to this file if you want the style definitions to take effect for only a particular skin. Information Console's JSPs typically link these styles in the following order:

- `<context root>\css\allstyles.css`

```
<LINK href="<html:rewrite page="/css/allstyles.css"/>"
      type="text/css" rel="stylesheet">
```
- `<context root>\iportal\activePortal\private\skins\<skin name>\css\skinstyles.css`

```
<LINK
  href="<ap:skinResource resource="/css/skinstyles.css" />"
  type="text/css" rel="stylesheet" >
```
- Style specifications from the customization web pages


```
<STYLE>
  <bean:write
    name="userinfo bean" property="skinConfig.cssCode" />
</STYLE>
```

If a style is defined in more than one of these files, the JSP engine uses the definition in the last file that contains the style. Thus the settings you specify in the customization web pages override any other CSS files.

`allstyles.css` contains additional style definitions for the Actuate Information Console application. Modify `allstyles.css` to change any style definitions that are not handled within the customization web pages or the `<context root>\iportal\activePortal\private\skins\<skinname>\css\skinstyles.css` file. Changes to a

style in allstyles.css affects all Information Console skins except the parameters page unless the customization web pages or a skin's skinstyles.css file override it. To customize the parameter component, modify the style definitions in the <context root>\css\parameter.css file.

How to test and modify styles depending on the browser type

- 1 Near the top of your JSP, link in the allstyles.css style sheet:

```
<LINK href="<html:rewrite page="/css/allstyles.css"/>"
      type="text/css" rel="stylesheet" >
```

- 2 After this line, link in the style sheet located in the current skin's css directory:

```
<LINK href="<ap:skinResource resource="/css/skinstyles.css" />"
      type="text/css" rel="stylesheet" >
```

- 3 Use the Jakarta Struts bean:write custom tag to generate and include style definitions for styles defined using the skin customization pages:

```
<STYLE>
  <bean:write
    name="userinfobean" property="skinConfig.cssCode" />
</STYLE>
```

- 4 If the skin customization styles contain any settings that do not work in a specific browser, you can override them individually.

Specifying colors and fonts

Specify fonts and colors for styles in the customization web pages or in the cascading style sheets. Specify colors using the following methods:

- Using a color name such as navy, yellow, or teal, as shown in the following example:

```
color: Yellow;
```
- Using hexadecimal notation to set the amount of red, green, and blue to use in the color.

```
#FFFF00
```
- Using decimal notation to set the amount of red, green, and blue to use in the color. In the customization web pages, fill in the value for red, green, and blue in the corresponding fields. In a CSS file, use a call to the rgb() method, as shown in the following example:

```
color: rgb(156, 207, 255);
```

How to change the font style of a single item

To change Actuate Information Console pages to display the user, system name, and volume in 12-point italic Comic Sans MS font:

1 In a text editor, open <context root>\css\allstyles.css.

2 Locate the following string:

```
bannerTextArea
```

There are two instances of the string bannerTextArea. The first is part of the definition for all the banner styles. This definition sets the banner styles' common attributes. The second instance sets the attributes for bannerTextArea only and looks like the following text:

```
.bannerTextArea {  
    color: white;  
    font-size: 10pt;  
    text-align: left;  
    white-space: nowrap;  
}
```

3 Modify the code that follows the bannerTextArea definition to change the font as shown in the following code:

```
.bannerTextArea {  
    color: white;  
    font-family: Comic Sans MS;  
    font-size: 11pt;  
    font-style: italic;  
    text-align: left;  
    white-space: nowrap;  
}
```

4 Save and close the CSS file.

5 Refresh your web browser to view the changes. Figure 2-23 shows the new appearance of the banner.



Figure 2-23 Appearance of customized Information Console banner

Customizing page styles for BIRT Studio

To customize BIRT Studio pages, use the files in <context root>\portal\bizRD\styles. This directory includes the following customizable CSS files:

- accordion.css defines styles for the report design area of the page, which displays the Available Data, Report Template Items, and other selectable tree views.
- dialog.css defines styles for dialog boxes that have shared characteristics, including the dialog boxes for template selection, file browsing, calculations, parameters, and so on.

- dialogbase.css defines the style of dialog containers, such as the button style, the Close icon style, and so on.
- title.css defines styles for the title bar of BIRT Studio pages.
- toolbar.css defines styles for the toolbar.
- wrcontextmenu.css defines the styles for BIRT Studio context menus.

Another file in this directory, webreporting.css, is not customizable.

For more information about using cascading style sheets, access the following URL:

<http://www.w3.org/Style/CSS/>

Modifying graphic images

Information Console pages use images for the company logo in the banners, on the side menu, and for the background. Some pages use additional images that are related to their content. You can also add new images on pages.

Certain images are most easily changed by customizing a skin. You can customize the company logo and the My Folder icon for all skins. In addition, you can customize the open and closed folder icons and volume icon for a skin that is cloned from the Treeview skin. These and all other images that you can customize reside in <context root>\portal\activePortal\private\skins\<skin name>\images. Update these images by using the skin customization pages to use new graphic files instead of changing the supplied graphic files. Customizing the images described in Table 2-10 affects most Information Console web pages.

Table 2-10 Images in Information Console skins

Skins	Default image file	Description
All	logo.gif	The company logo to use in the banners
All	homefoldericon.gif	The image to use beside the My Folder link
Treeview	closedfoldericon.gif	The image to use to indicate a unexpanded folder in the hierarchical view of the volume and folders
Treeview	foldericon.gif	The image to use to indicate an expanded folder in the hierarchical view of the volume and folders
Treeview	volume_icon.gif	The image to use to indicate a volume in the hierarchical view of the volume and folders

An additional image of interest is <context root>\portal\activePortal\private\skins\<skin name>\images\background.gif. The Classic skin and its clones use this image to provide the background for every page. This image is one pixel high

and 1280 pixels long, and is copied as necessary to fill the page. Change the contents of this image file to modify the background of a Classic skin clone.

All other images reside in <context root>\iportal\activePortal\images. This set of images provides the features on the side menu in the Classic skin and the tree in the Treeview skin. Update these feature images by changing the corresponding feature definition in the \iportal\WEB-INF\functionality-level.config file.

Other images are referenced by hard-coded path and file names in JSP and JavaScript files, such as the icons in <context root>\iportal\activePortal\private\filefolders\views\categories.jsp. For example, categories.jsp specifies the location and filename, <context root>\iportal\activePortal\images\detailicon.gif, a magnifying glass icon that is used to obtain more details about a document or other item in a list. When you change the location or replace an image with a new file, you must update the JavaScript and JSP files that use them. Alternatively, make a backup copy of the original image and then reuse the original name for your new image. By reusing the original name, you do not need to make any changes in the JSP and JavaScript files using the image.

How to replace the detail icon with your own icon

Actuate Information Console uses a magnifying glass icon to display more information about files, channels, and jobs. For example, <context root>\iportal\activePortal\private\jobs\completedjob.jsp contains the following code using this image:

```
"
    border="0" align="middle"
    alt="<bean:message bundle="iportalResources"
    key="TTIP_JOB_DETAIL"/>"
    title="<bean:message bundle="iportalResources"
    key="TTIP_JOB_DETAIL"/>" />
```

- 1 Create your new details image in <context root>\iportal\activePortal\images. The default Actuate Information Console icon, detailicon.gif, is 12 pixels by 13 pixels. During development, use a new name, such as new_detailicon.gif.
- 2 Rename the existing details image, <context root>\iportal\activePortal\images\detailicon.gif, to another file name, such as detailicon_original.gif.
- 3 Rename your new details image to detailicon.gif.
- 4 Close your browser, re-open Information Console, and log in. The new detail icon appears in all places that Actuate Information Console had displayed the magnifying glass icon. In Figure 2-24, the default detailicon.gif image has been replaced by an image of a question mark.

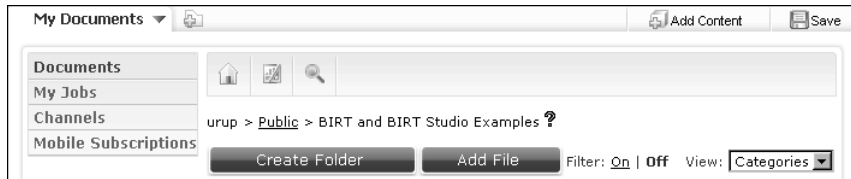


Figure 2-24 Customized skin with modified detail icon

If you want to replace only some instances of `detailicon.gif`, search the files in the context root for all files that use that image. Then replace that file name with your image's file name in only some of the files. For example, you could use the default magnifying glass in most places but change `<context root>\portal\activePortal\private\common\breadcrumb.jsp` to use your own image.

Follow similar procedures to customize other images in Actuate Information Console pages that are not specified in the skin manager or in `<context root>\WEB-INF\functionality-level.config`.

Part Two

Actuate Information Console reference

Actuate Information Console configuration

This chapter contains the following topics:

- About Information Console configuration
- Configuring the Information Console web application
- Configuring the connection to iServer
- Configuring Actuate Analytics
- Configuring the BIRT Viewer and Interactive Viewer
- Configuring BIRT Studio
- Configuring BIRT Data Analyzer

About Information Console configuration

The Information Console application is configured using files in the context root's WEB-INF directory. For example, the web.xml configuration file for your context root is located:

```
<context root>\WEB-INF\web.xml
```

Table 3-1 lists the configuration files discussed in this chapter.

Table 3-1 Information Console configuration files

File	Features	Description
erni_config.xml	BIRT Studio	Configures BIRT Studio functionality
experience.levels	Information Console, Actuate Analytics Cube Viewer	Configures the Actuate Analytics Experience Levels for Information Console
functionality-level.config	Information Console	Configures the Information Console user interface by iServer security roles
iv_config.xml	BIRT Viewer	Configures BIRT Viewer user interface
localemap.xml	All	Configures languages and locales
TimeZones.xml	All	Configures time zones
volumeProfile.xml	All	Consolidates iServer volume connection information into a single handle, hiding iServer volume details in a URL
web.xml	All	Configures features of the Information Console including security, networking, caching, labeling and storage

Configuring the Information Console web application

The Information Console provides the ability to organize, run, and view reports. You configure the user interface, logging, and caching for the Information Console using web.xml.

Configuring the Information Console using web.xml

Web.xml contains parameters that control Information Console features. Table 3-2 describes the configuration parameters for the Information Console application.

Table 3-2 Actuate Information Console web.xml parameters

Parameter name	Description
BIRT_RENDER_FORMAT_EMITTER_ID_MAPPING	<p>Specifies which emitter will be used for a specific BIRT report. Valid entries are of the format "render_format:emitter_ID" separated by a semicolon. The default value is:</p> <p>html:org.eclipse.birt.report.engine.emitter.html;xhtml:com.actuate.birt.report.engine.emitter.xhtml;pdf:org.eclipse.birt.report.engine.emitter.pdf;postscript:org.eclipse.birt.report.engine.emitter.postscript;xls:com.actuate.birt.report.engine.emitter.xls;ppt:org.eclipse.birt.report.engine.emitter.ppt;pptx:com.actuate.birt.report.engine.emitter.pptx;doc:org.eclipse.birt.report.engine.emitter.word;docx:com.actuate.birt.report.engine.emitter.docx</p>
CACHE_CONTROL	<p>Specifies how a web browser caches information using one of the following values:</p> <ul style="list-style-type: none">■ NO-CACHE indicates that the browser does not cache information and forwards all requests to the server. With NO-CACHE, the back and forward buttons in a browser do not always produce expected results, because choosing these buttons always reloads the page from the server. If multiple users access Information Console from the same machine, they can view the same cached data. Setting CACHE_CONTROL to NO-CACHE prevents different users viewing data cached by the browser.■ NO-STORE indicates that information is cached but not archived.■ PRIVATE indicates that the information is for a single user and that only a private cache can cache this information. A proxy server does not cache a page with this setting.■ PUBLIC indicates that information may be cached, even if it would normally be non-cacheable or cacheable only within an unshared cache.■ Unset (no value) is the default value. The browser uses its own default setting when there is no CACHE_CONTROL value. <p>Caching information reduces the number of server requests that the browser must make and the frequency of expired page messages. Caching increases security risks because of the availability of information in the cache. For additional information about cache control, see the HTTP/1.1 specifications.</p>

(continues)

Table 3-2 Actuate Information Console web.xml parameters (continued)

Parameter name	Description
CONNECTION_TIMEOUT	Controls how many seconds Actuate Information Console waits for a request to complete before dropping the connection to the application server or Actuate BIRT iServer. Set this value to limit wait times. The default value is 0, meaning the connection is never dropped.
COOKIE_DOMAIN	Specifies the host name of the server setting the cookie. The cookie is only sent to hosts in the specified domain of that host. The value must be the same domain the client accesses. Information Console automatically sets this parameter. For example, if the client accesses <code>http://www.actuate.com/iportal/login.do</code> , the domain name is <code>actuate.com</code> .
COOKIE_ENABLED	Indicates whether to use cookies to store information between user logins. The default value is True. If False, Information Console does not use cookies. Without cookies, many Information Console features are unavailable or do not persist across sessions. For example, without cookies, user name, language, and time zone settings always use their default values when a new browser session begins.
COOKIE_SECURE	Indicates whether to access and write cookies securely. If true, cookies are only written if a secure connection, such as HTTPS, is established. The default value is false, which enables cookies for all connection types.
DEFAULT_ESS_VIEWING_FORMAT	Specifies the default format for viewing spreadsheet reports. Valid values include XLS, XLSX, and PDF. The default value is XLS.
DEFAULT_LOCALE	Specifies the default locale. Information Console sets this parameter value during installation. The locale map is <code><context root>\WEB-INF\localemap.xml</code> .
DEFAULT_PAGE_BREAK_INTERVAL	Specifies the number of rows to display in one page when viewing a report. If set to 0, there are no page breaks.
DEFAULT_TIMEZONE	Specifies the default time zone. Information Console sets this parameter value during installation. The time zone map is <code><context root>\WEB-INF\TimeZones.xml</code> .
MOBILE_APP_DOWNLOAD	The URL target of the download button displayed by Information Console when viewed in mobile/touch device.
ENABLE_CLIENT_SIDE_REDIRECT	Specifies whether URL redirection is done on the client side or the server side. Set the value to True for client side redirection. The default value is False. For more information about URL redirection, see “Using proxy servers with Actuate Information Console,” in Chapter 1, “Introducing Actuate Information Console.”

Table 3-2 Actuate Information Console web.xml parameters (continued)

Parameter name	Description
ENABLE_DEBUG_LOGGING	Indicates whether to record debugging messages in a log file called Debug.log. Set the value to True to enable debug messages in the log file. The default value is False.
ENABLE_ERROR_LOGGING	Indicates whether to log errors. This parameter's default value is True, which enables error logging. If you set this parameter to True, Information Console creates two error log files: <ul style="list-style-type: none"> ■ Admin.log records general errors. ■ Soapfault.log records iServer communication errors.
ENABLE_JUL_LOG	Indicates whether to log Information Console activity. This parameter's default value is TRUE, which enables logging. If you set this parameter to TRUE, Information Console creates log files named reportService.<Service number>.<System name>.<Information Console start up time stamp>.<File number>.log.
ERROR_LOG_FILE_ROLLOVER	Specifies the time period to wait before starting a new log file. Options are Daily, Monthly, Weekly, and Yearly. The default value is Monthly.
EXECUTE_DASHBOARD_GADGET_GENERATION_WAIT_TIME	Specifies the time to wait, in seconds, for a gadget to generate when running a dashboard design file. This parameter's default value is 2 seconds.
EXECUTE_REPORT_WAIT_TIME	Specifies the time to wait, in seconds, for a report to execute. This parameter's default value is 20 seconds. For more information about the wait time parameter, see "execute report page," and "execute page," in Chapter 4, "Actuate Information Console URIs."
FILES_DEFAULT_VIEW	Specifies the default view for the files and folders list using one of the following values: <ul style="list-style-type: none"> ■ Categories, the default, displays files organized in rows by type. ■ Detail displays files organized in rows by name. ■ List displays files organized in columns with small icons. ■ Icon displays files organized in columns with large icons.
FORCED_GC_INTERVAL	Indicates the length in seconds of the interval that the Information Console application waits between forced garbage collections. To disable garbage collection, set this parameter to 0, the default value. Use this parameter to tune application server performance. 600 seconds is the recommended value. If the value is too low, the application server performs garbage collection too frequently, slowing the system. If the value is too high, you waste memory. If disabled, the application server controls garbage collection.

(continues)

Table 3-2 Actuate Information Console web.xml parameters (continued)

Parameter name	Description
GADGET_GENERATION_WAITING_TIME	Specifies the time to wait, in seconds, for an individual gadget to generate. This parameter's default value is 10 seconds.
IDAPI_TIMEOUT	Specifies the number of seconds to wait for a SOAP message response. This value must be larger than the maximum time necessary to run a report design or Information Console generates a time-out error for some reports. Its default value is 7200.
INSTALL_MODE	Indicates whether Information Console is installed with iServer. The value is set when Actuate Information Console is installed. Do not change this setting.
JUL_LOG_CONSOLE_LEVEL	The level of Information Console activity to log to the console. Valid values are OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, in order of the number of messages to log. The default value is OFF.
JUL_LOG_FILE_COUNT	Specifies the number of log files for a particular time stamp, if the value of ENABLE_JUL_LOG is TRUE.
JUL_LOG_FILE_LEVEL	The level of Information Console activity to log in a file. Valid values are OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, in order of the number of messages to log. The default value is WARNING.
JUL_LOG_FILE_SIZE_KB	The maximum size, in kilobytes, for an Information Console activity log file. When a log file reaches this size, Information Console creates a new log file and increments its file number. If the log file number reaches the value of JUL_LOG_FILE_COUNT, Information Console resets the file number to zero and overwrites the first log file for the time stamp.
LOG_FILE_LOCATION	Indicates which directory contains the log files. If the value is not an absolute directory path name, Actuate Information Console locates the directory in the Information Console home directory. The default value is logs in the Information Console home directory.
LOGIN_TIMEOUT	Specifies the number of seconds to wait before a session times out. The minimum login time-out is 300 seconds. The maximum value is equivalent to java.lang.Long. Its default value is 1200 seconds.
MAX_BACKUP_ERROR_LOGS	Specifies the maximum number of backup error log files to keep. The default value is 10.
MAX_LIST_SIZE	Limits the number of items returned when getting folder items, jobs, job notices, scheduled jobs, and channels to reduce network traffic. The default value is 150.

Table 3-2 Actuate Information Console web.xml parameters (continued)

Parameter name	Description
PRELOAD_ENGINE_LIST	List of engines to load when Information Console starts. Valid values are birt and ess. Default value is "birt, ess" which indicates both.
PROGRESSIVE_REFRESH	Controls the interval in seconds at which an Actuate report refreshes itself when running a progressive report. The report refreshes first after 15 seconds, then after 60 seconds, and then after the PROGRESSIVE_REFRESH interval. If the value is less than 60, Actuate Information Console uses 60 seconds. This parameter's default value is 1800 seconds.
PROGRESSIVE_VIEWING_ENABLED	Specifies whether a paginated report starts to display in the browser as soon as the first page has been generated. Valid values are true and false. The default value is true.
PROXY_BASEURL	Indicates a proxy server's URL if the network uses one between Information Console and the client. The default value is blank, which indicates that the network does not use a proxy server.
SECURITY_ADAPTER_CLASS	Specifies the fully qualified class of the security adapter, which must extend com.actuate.iportal.security.iPortalSecurityAdapter, that controls access to Actuate Information Console functionality. The default value is no name.
SESSION_DEFAULT_PARAMETER_VALUE_ID	Specifies the name of the object that stores the HTTP session-level report parameters. This object is an instance of the com.actuate.parameter.SessionLevelParameter class, which is extensible. The default value is SessionDefaultParameterValue.
sessionTimeout	The number of milliseconds the Information Console Ajax Proxy maintains an idle session. The default value is 5000.
TRANSIENT_STORE_MAX_SIZE_KB	Limits the amount of disk space that Actuate Information Console uses for temporary files. The default value is 102400, which is 100 MB.
TRANSIENT_STORE_PATH	Path to Actuate Information Console transient files. The default value is set when Information Console is installed. When deploying more than one context root or separate server, set a unique path for each.
TRANSIENT_STORE_TIMEOUT_MIN	Specifies, in minutes, how long to retain Actuate Information Console transient files. The default value is 40, which is 40 minutes.
UPLOAD_FILE_TYPE_LIST	Specifies the valid file types, by extension, for upload with Information Console. The default value is blank, which indicates any file type.

(continues)

Table 3-2 Actuate Information Console web.xml parameters (continued)

Parameter name	Description
UPLOAD_SECURITY_MANAGER	Specifies the fully qualified class of the security adapter, which must extend com.actuate.iportal.security.IUploadSecurityAdapter, that controls access to the upload functionality. The default value is no name.
VIEW_XLS_IN_REQUESTER	Indicates that a spreadsheet report in Excel format always opens in the same browser as Information Console. The default value is false, indicating that Excel files open in a separate window.

Configuring Information Console using volumeProfile.xml

Information Console uses a volume profile to access a specific iServer instance and Encyclopedia volume. Because the volume profile conceals the iServer and Encyclopedia volume values from the users of Information Console, the system administrator can change the location of these resources without affecting the URLs accessed by the users. To access iServer resources using a volume profile, add a `__vp=ProfileName` parameter to the URL.

Customize volume profiles by creating or modifying entries in the following file:

```
<context root>\WEB-INF\volumeProfile.xml
```

For example, the following is a volume profile definition for the server1 server:

```
<VolumeProfiles>
  <Profile>
    <Default>true</Default>
    <ProfileName>server1</ProfileName>
    <RepositoryType>enterprise</RepositoryType>
    <ServerUrl>http://server1:8000</ServerUrl>
    <Volume>volume1</Volume>
    <DashboardTemplatePath></DashboardTemplatePath>
  </Profile>
</VolumeProfiles>
```

- `<ProfileName>` is the name of this profile.
- `<RepositoryType>` has one of two values, either enterprise or workgroup.
- `<ServerUrl>` contains the iServer URL, for example, `http://server1:8000`. If repositorytype is workgroup, ServerUrl is ignored.
- `<volume>` is the volume name. If repositoryType is workgroup, volume is ignored.
- `<Default>` is optional. Valid values are true and false. A value of true sets this profile as the default volume profile, and the server and volume in this profile

is used if no volume profile is provided in the URL. Information Console handles only the first profile with default set to true as the default profile.

- `<DashboardTemplatePath>` is optional. This repository path is the location of the dashboard file that loads when a user creates a new dashboards.

To make a new profile available to Information Console, add a new `<Profile>` element to the list in `<VolumeProfiles>` in `volumeProfile.xml`. Then, restart the Information Console. For example, the following profile accesses the `volume2` volume on the `server2` server:

```
<Profile>
  <Default>false</Default>
  <ProfileName>server2</ProfileName>
  <RepositoryType>enterprise</RepositoryType>
  <ServerUrl>http://server2:8000</ServerUrl>
  <Volume>volume2</Volume>
  <DashboardTemplatePath></DashboardTemplatePath>
</Profile>
```

Using a volume profile defined in `volumeProfile.xml`

Information Console connects to the server and volume defined by the default volume profile entry when the URL does not include the `__vp` parameter or the volume parameter. For example, to connect to the default volume and server, use the following URL:

```
http://infoconsole:8900/iportal/getfolderitems.do?userid=userName
&password=validPassword
```

Information Console connects to the server and volume defined by a volume profile when the URL contains a `__vp` parameter with a valid profile name and the URL does not have a volume parameter. For example, to connect to `volume2` on `server2` defined by the volume profile example above, use the following URL:

```
http://infoconsole:8900/iportal/getfolderitems.do?userid=userName
&password=validPassword&__vp=server2
```

Overriding the volume specified in a volume profile

Information Console supports using a volume profile to access an Encyclopedia volume even if the volume is not specified in a volume profile definition. To override the volume specified by a volume profile, add the volume parameter to the URL. A URL with a valid `__vp` parameter and volume parameter connects to the server in the volume profile, but the volume assigned to the volume parameter. For example, to connect to `volume3` on `server2` explicitly, use the following URL:

```
http://infoconsole:8900/iportal/getfolderitems.do?userid=userName
&password=validPassword&__vp=server2&volume=volume3
```

If the URL contains a volume parameter but not a __vp parameter, Information Console connects to the server in the default volume profile and the volume assigned to the volume parameter. For example, to connect to volume3 on the default server, use the following URL:

```
http://infoconsole:8900/getfolderitems.do?userid=userName
&password=validPassword&volume=volume3
```

Understanding temporary volume profiles

If a request URL contains serverurl, repositorytype, or volume parameters not defined in volumeProfile.xml, Information Console generates a temporary profile name for this set of volume properties. A temporary name is not persistent and is lost every time the application restarts. If the request URL does not contain serverurl, volume, and repositorytype parameters, Information Console uses the default profile for the request URL. If there is no default profile defined, Information Console generates a temporary server profile having a random name and uses SERVER_DEFAULT, DEFAULT_VOLUME, and REPOSITORY_TYPE defined in WEB-INF/web.xml as the default values for serverurl, volume, and repositorytype.

Configuring Information Console functionality levels with functionality-level.config

A functionality level defines which Information Console user interface features are visible and usable by members of an Encyclopedia volume security role or roles. By default, every user can access all functionality levels. Functionality levels corresponding to iServer security roles like Intermediate, Advanced, and Administrator, are provided in the comments of the functionality-level.config file. For example, by default every functionality level shows Log out, Options, and Help links on the Information Console banner.

The Intermediate and Advanced levels add a Search link to the documents page and the capability to add tabs, and the Administrator level adds a Customization link, as shown in Figure 3-1.

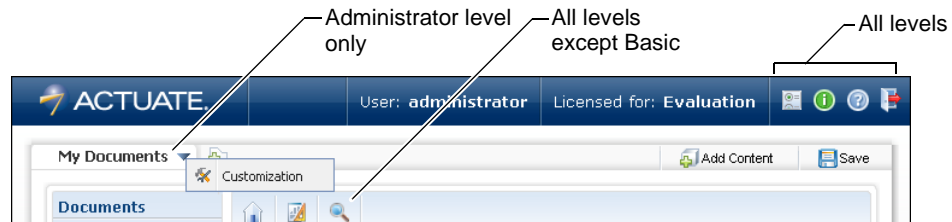


Figure 3-1 The banner appearance for a user at the Administrator functionality level

Actuate Information Console provides five functionality levels by default. Four functionality definitions specify a corresponding Encyclopedia volume security

role that provides access to that functionality level. Table 3-3 shows the functionality levels and their corresponding security roles. The Administrator level is the Information Console Administrator, not the Encyclopedia volume administrator.

Table 3-3 Information Console default functionality levels and the corresponding Encyclopedia volume security roles

Functionality level	Security role
Basic	All (The All role includes all users.)
Intermediate	Active Portal Intermediate
Advanced	Active Portal Advanced
Administrator	Active Portal Administrator

Customize a functionality level by creating or modifying entries in the following file:

```
<context root>\WEB-INF\functionality-level.config
```

When modifying the configuration file, ensure that functionality levels in the configuration file specify a corresponding security role to enable access to that functionality level. You can modify the built-in levels but you cannot delete them.

The following example shows the definition of the Basic functionality level:

```
<Level>
  <Name>Basic</Name>
  <Role>All</Role>
  <FeatureID>Jobs</FeatureID>
  <FeatureID>Documents</FeatureID>
  <FeatureID>Channels</FeatureID>
  <SubfeatureID>DeleteFile</SubfeatureID>
  <SubfeatureID>InteractiveViewing</SubfeatureID>
  <AnalyticsExperienceLevel>Novice</AnalyticsExperienceLevel>
  <AnalyticsExperienceLevel>Standard</AnalyticsExperienceLevel>
  <AnalyticsExperienceLevel>Advanced</AnalyticsExperienceLevel>
</Level>
```

Every functionality level entry in the configuration file must have the five components shown in the following sections.

Name

Use a unique alphanumeric string for the functionality level name, enclosed within the <Name> and </Name> tags, such as <Name>Intermediate</Name>.

Role

The Role component defines the name of the Encyclopedia volume security role that corresponds to the functionality level. Both the security role and the functionality level must exist before you can assign the functionality level to a

user. Enclose the security role name within <Role> and </Role> tags, such as <Role>Active Portal Intermediate</Role>.

Features

There are five features, which are described in Table 3-4.

Table 3-4 Features of functionality levels

Feature	Description
Channels	Provides access to channels
Customization	Provides access to skin customization
Documents	Provides access to files and folders
Jobs	Supports submitting and accessing jobs
Mobile	Provides access to BIRT mobile viewing
Search	Provides access to file and folder search

Enclose the feature within <FeatureID> and </FeatureID> tags. When you omit a feature from a functionality level, the corresponding side menu or banner item is not visible to anyone using that functionality level. For example, the Search feature is not available to the Basic functionality level, so the Search link does not appear in the banner for a user at the Basic functionality level.

Feature IDs

Functionality-level.config defines the features that are available to Information Console users as well as functionality levels. The following example shows the Documents feature definition from functionality-level.config:

```
<Feature>
  <ID>Documents</ID><Labelkey>SBAR_DOCUMENTS</Labelkey><Link>
    /getfolderitems.do</Link>
  <SmallIcon>/iportal/activePortal/images/
    filesfoldersicon16x16.gif
  </SmallIcon>
  <LargeIcon>/iportal/activePortal/images/filesfoldersicon.gif
  </LargeIcon>
</Feature>
```

The ID identifies the feature for Information Console. The label key appears on the side menu for Documents, Jobs, and Channels, or in the banner for Search and Customization. The link specifies the action that is executed for the feature. The small and large icons represent the feature in the side menu. Only the side menu features use the small and large icons.

Although you can customize the labels and links of all five features, do not change the <ID> or <Labelkey> tag values. Information Console uses these tags to identify the features and perform resource management. The Labelkey provides the resource to use for the feature's text label.

Changing the Link tag’s value specifies a different action to execute. Changing the icon files changes the side menu’s appearance. The small icons are used by the Treeview skin and are 16x16 pixels. The large icons are used by the Classic skin and are 32x32 pixels. The Tabbed skin does not use icons. Link and icon file names are relative to <context root>.

Subfeatures

A subfeature corresponds to an action you can perform using the Information Console user interface. A user must have appropriate privileges to create, delete, or share files or folders. Table 3-5 describes the subfeatures.

Table 3-5 Subfeatures of the features described in Table 3-4

Feature	Subfeature	Supported functionality
Channels	SubscribeChannel	Subscribing to channels.
Documents	AddFile	Uploading files.
Documents	CreateFolder	Creating folders.
Documents	DeleteFile	Deleting files.
Documents	DeleteFolder	Deleting folders.
Documents	DownloadFile	Downloading files.
Documents	ShareFile	Sharing files.
Jobs	JobPriority	Setting job priority, up to the user’s maximum job priority.
Jobs	SelfNotification WithAttachment	E-mail notification for successful jobs.
None	InteractiveViewing	Using BIRT Interactive Viewer.
None	AdvancedData	Used in BIRT Studio.
None	DashboardBusiness User	Viewing and editing dashboards and gadgets.
None	DashboardDeveloper	Creating and configuring gadgets and dashboards.
None	ShareDashboard	Sharing dashboards. Requires either DashboardBusinessUser or DashboardDeveloper.

Specify one subfeature to a line and enclose each subfeature within <SubfeatureID> and </SubfeatureID> tags. Each subfeature is associated with a feature. You cannot include a subfeature in a functionality level if its corresponding feature is not available to that functionality level.

Analytics experience levels

Analytics experience levels is a list of Actuate Analytics experience levels available to users at the current functionality level. The default behavior is that all experience levels are available at all functionality levels. Users can select their

own Actuate Analytics user experience level on the Information Console Options page or from the Actuate Analytics Cube Viewer. Enclose each experience level within `<AnalyticsExperienceLevel>` and `</AnalyticsExperienceLevel>` tags. You can use multiple experience level tags. For more information about experience levels, see “Configuring experience levels for Actuate Analytics Cube Viewer,” later in this chapter.

Configuring Information Console locales

`<context root>\WEB-INF\localemap.xml` contains the locales available to Information Console. Add locales to this file by using the same format as the existing locales. To see each locale in the file, search for one of the following strings:

`<Locale`

or:

`<DisplayName>`

Searching for `<Locale` places the cursor on the line having the ID for the locale. Searching for `<DisplayName>` places the cursor on the line having the descriptive name for the locale.

Typically, the locale names have the following syntax:

`<language>_<country>`

For example, `ar_EG` is Arabic (Egypt). A language spoken in multiple countries has multiple locale names for which the language code is the same and the country code has several values. For example, `en_US` is the locale for English (United States), `en_AU` is the locale for English (Australia), and `en_BZ` is the locale for English (Belize). Some countries have several locales, one for each language. For example, Canada has both `en_CA` for English (Canada) and `fr_CA` for French (Canada). You specify a default locale for a custom web application in `<context root>\WEB-INF\web.xml`.

Configuring Information Console time zones

`<context root>\WEB-INF\TimeZones.xml` contains the time zones available to Information Console. Add time zones to this file by using the same format as the existing time zones. To see each time zone in the file, search for one of the following strings:

`<TimeZone`

or:

`<DisplayName>`

Searching for <TimeZone places the cursor on the line having the ID for the time zone. Searching for <DisplayName> places the cursor on the line having the descriptive name for the time zone.

Some time zone names have short abbreviations for the ID. All time zone names have a full descriptive ID, such as Samoa Standard Time or Greenwich Standard Time. The DisplayName provides the relative time from Greenwich Standard Time and one or more locations that the time zone includes. You specify a default time zone for a custom web application in <context root>\WEB-INF\web.xml.

Customizing messages and text according to locale

Error messages and text for Information Console are encoded in resource files compressed in the <context root>/WEB-INF/lib/resources.jar file. The properties files contain entries for the interface text and error codes Information Console generates.

For reference, the <context root>/WEB-INF/ErrorMessage.txt file lists the default error codes used by Information Console. The \com\actuate\reportcast\resources\ErrorMessages.properties file within the resources.jar archive contains error messages for the default locale. Information Console uses messages from this file if no locale-specific message for the error exists. Not all of the codes exist in the default ErrorMessages.properties because iServer directly generates many of them in the SOAP messages sent to Information Console.

Override iServer and Information Console messages using a locale-specific error messages file. In addition to the default ErrorMessages.properties file, Information Console provides several localized error message files, such as ErrorMessages_de_DE.properties. This file contains the German language messages for the Germany locale. To specify error messages to a certain locale, modify the existing error message file for that locale or create a new file for the locale. By convention, the format of a locale-specific error message file name includes the language and locale codes at the end of the file name separated by underscore characters.

For example:

```
ErrorMessages_de_DE.properties
```

- de is the language code for German.
- DE is the Germany country code.

These values for language and locale codes are defined in localemap.xml.

Because alphabets for different languages are dissimilar and Information Console uses ASCII encoding for these files, you must convert new or edited files into ASCII format. To convert the files to ASCII, modify the properties file using an editor that saves to the UTF-8 format and convert the file to ASCII using the Java native2ascii utility using the -encoding UTF-8 switch. The native2ascii utility

installs with any Java Developer Kit in the <JDK home>/bin directory. Model the format of new messages after those in the ErrorMessage.properties file.

When your modifications are complete, recompress the resources.jar archive using the Java jar utility, retaining the original directory structure for the archive. Copy the new resources.jar file to the <context root>/WEB-INF/lib directory, restart the Actuate 11 Apache Tomcat for Information Console service, and log in using the locale for the modified messages file. Confirm that the new messages file was loaded by examining the error messages generated by Information Console using that specific locale.

Error messages appear in pop-up windows when an error is encountered. The window is an operating system window, not an HTML frame. If you use a language-specific version of Windows corresponding to the locale you are viewing, the localized message shows up correctly. If you have not loaded the Windows language pack for a language, the text of a message appears as empty squares instead of text.

Configuring Shindig 2.0 for a WAR or EAR deployment

To enable shindig 2.0 support for a deployable Information Console's WAR or EAR file, modify the host name, port, and context root in the following configuration files:

```
<context root>/WEB-INF/web.xml
<context root>/WEB-INF/classes/shindig.properties
<context root>/WEB-INF/classes/containers/default/container.js
```

In web.xml, update the following parameters:

```
<param-name>system.properties</param-name>
<param-value>
    shindig.host=<host name>
    shindig.port=<port>
</param-value>
```

- <host name> is the name of the web server hosting the Information Console web application.
- <port> is the TCP port assigned to the Information Console web application

In shindig.config, update the following parameter:

```
shindig.signing.global-callback-url=http://<host name>:<port>/
<context>/gadgets/oauthcallback
```

- <host name> is the name of the web server hosting the Information Console web application.
- <port> is the TCP port assigned to the Information Console web application
- <context> is the context root of the Information Console web application.

In container.js update the following parameters:

```
"gadgets.jsUriTemplate" : "http://%host%/<context>/gadgets/js/
%js%"
"gadgets.oauthGadgetCallbackTemplate" : "://%host%/<context>/
gadgets/oauthcallback"
"gadgets.osDataUri" : "http://%host%/<context>/social/rpc"
"proxyUrl" : "://%host%/<context>/gadgets/
proxy?refresh=%refresh%&url=%url%",
"jsonProxyUrl" : "://%host%/<context>/gadgets/makeRequest"
"path" : "http://%host%/<context>/social"
"endPoints" : [ "http://%host%/<context>/social/rpc", "http://
%host%/<context>/gadgets/api/rpc" ]
```

<context> is the context root of the Information Console web application.

Configuring the connection to iServer

The Information Console provides the ability to connect to iServer, an Encyclopedia volume, and manage reports on remote systems. Configure the repository, network, and Message Distribution service for the Information Console using parameters in web.xml. These parameters control the Information Console’s connection to iServer and the Encyclopedia volume. Table 3-6 describes the configuration parameters for networking with iServer.

Table 3-6 iServer connection web.xml parameters

Parameter name	Description
AUTO_SAVE_DASHBOARD_DELAY	Controls how long, in seconds, the dashboard engine should wait before sending a save message to persist the personal dashboard file. To disable auto save, set this value to 0.
DASHBOARD_SHARED_RESOURCES	Specifies the path for the shared dashboard and gadget resources on the Encyclopedia volume. The gadget gallery displays the contents of this folder under the shared folder and is the default location when sharing dashboards.
MAX_CONNECTIONS_PER_SERVER	Indicates the maximum number of Actuate Information Console connections to Actuate BIRT iServer. Actuate pools connections to increase efficiency. Choose a number of connections that satisfies the most requests concurrently without requiring an unreasonable amount of memory. Begin with a value equal to the number of threads available in your application server. The value for this parameter must be greater than 0. The default value is 150.

(continues)

Table 3-6 iServer connection web.xml parameters (continued)

Parameter name	Description
MDS_ENABLED	Indicates whether to enable the Message Distribution service. The default value is True, which enables the Message Distribution service. For more information about the Message Distribution service, see “Understanding Actuate Information Console load balancing,” in Chapter 1, “Introducing Actuate Information Console.”
MDS_REFRESH_FREQUENCY_SECONDS	Indicates, in seconds, how quickly Actuate Information Console detects an offline or new node in a cluster. If MDS_ENABLED is True, Information Console refreshes the list of available nodes from Actuate BIRT iServer at the time interval specified. The default value is 300 seconds.
REPOSITORY_CACHE_TIMEOUT_SEC	Specifies how long a repository cache remains valid. When the cache becomes invalid, any user actions refresh the cache for the time-out duration. The default value is 900 seconds.
TEMP_FOLDER_LOCATION	Specifies the directory the Information Console uses to temporarily store files from an Encyclopedia volume if viewing the file requires a location on the web server. If the value is not an absolute directory path name, Actuate Information Console locates the directory in the Information Console home directory. The default value is temp in the Information Console home directory. The Information Console user must have write permission for the directory. When deploying more than one context root or separate server, set a unique path for each.
VOLUME_PROFILE_LOCATION	Path to the volume profile configuration file from the context root. Default value is /WEB-INF/VolumeProfile.xml.

Configuring Actuate Analytics

The Actuate Analytics option displays detailed information about report data in various formats. It can be configured to change its behavior and performance.

Configuring Actuate Analytics Cube Viewer

Parameters in web.xml control the operation of the Actuate e.Analysis option for e.reports. Table 3-7 describes the configuration parameters for Actuate Analytics.

Table 3-7 Actuate Analytics web.xml parameters

Parameter name	Description
ANALYTICS_BASE_EXPLEVEL_NAME	The experience level for the Actuate Analytics Cube Viewer if no experience level is assigned in the user's functionality level. The default is Novice.
ANALYTICS_CUBE_VIEW_RECORDS	Enables the drill to detail functionality in the Actuate Analytics Cube Viewer. Overrides the setting in the cube design. True to enable, False to disable. False is the default.
ANALYTICS_CUBE_VIEWER_HEIGHT	Height of the Actuate Analytics Cube Viewer application. The default is 100%.
ANALYTICS_CUBE_VIEWER_WIDTH	Width of the Actuate Analytics Cube Viewer application. The default is 100%.
ANALYTICS_ENABLE_ONETIME_DOWNLOAD	True to enable a one-time Actuate Analytics Cube Viewer download, False otherwise. This parameter only applies to the Microsoft Internet Explorer browser. The default is True.
ANALYTICS_ENABLE_SAVE_VIEW	True to enable or False to disable saving cube views to the Actuate BIRT iServer. The default is True.
CATEGORY_<CategoryName>	True to specify that Actuate Analytics Cube Viewer displays this category in the cube view or False to not display this category in the cube view. The default is True.
MEASURE_<MeasureName>	True to specify that Actuate Analytics Cube Viewer displays this measure in the cube view or False to not display this measure in the cube view. The default value is True.

Configuring experience levels for Actuate Analytics Cube Viewer

Parameters in experience.levels control user access to the Actuate Analytics Cube Viewer features. The Administrator can modify these levels and create new levels using the <context root>\WEB-INF\experience.levels file. If the experience.levels file is missing, all functionality is available to all users. The experience.levels file contains unicode strings, so it must be edited using a unicode compliant editor.

Experience levels control visibility of the toolbars in the Actuate Analytics Cube Viewer, the sets of visible buttons, and context menu items. Users select the experience level that most closely matches their expertise. Each experience level incorporates more features than the level before. Table 3-8 describes the default

functionality levels.

Table 3-8 Actuate Analytics Cube Viewer experience levels

Experience level	Description
Novice	Cube Viewer features for beginning users, including undo and redo, printing, expanding and collapsing views, using predefined views, and collaboration
Standard	Cube Viewer features for intermediate users, including all the Novice level features plus line charts, creating and deleting predefined views, categories, and filtering
Advanced	Cube Viewer features for advanced users, including all the Standard level features plus functions, viewing raw data, and view manipulation

The user's functionality level determines the experience levels the user can choose. Each functionality level description includes a list of experience levels available for that functionality level. Each experience level is listed within `<AnalyticsExperienceLevel>` and `</AnalyticsExperienceLevel>` tags. The default setting is that the only experience levels available are the experience levels defined in the functionality level. The following example shows the experience level entries:

```
<AnalyticsExperienceLevel>Novice</AnalyticsExperienceLevel>
<AnalyticsExperienceLevel>Standard</AnalyticsExperienceLevel>
<AnalyticsExperienceLevel>Advanced</AnalyticsExperienceLevel>
```

Each experience level is defined by a skin name, one or more display names, and a list of features that the experience level hides. When you customize an experience level, you can hide additional features or you can make additional features available to the user. You customize experience levels in the `<context root>\WEB-INF\experience.levels` file. The following code example shows elements of the Standard functionality level:

```
<EXPERIENCE_LEVELS xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <NUMBER_OF_LEVELS>3</NUMBER_OF_LEVELS>
  <DEFAULT_EXPERIENCE_LEVEL>2</DEFAULT_EXPERIENCE_LEVEL>
  ...
  <EXPERIENCE_LEVEL>
    <SKIN_NAME>Standard</SKIN_NAME>
    <DISPLAY_NAME>
      ...
      <NAME>Standard</NAME>
    </DISPLAY_NAME>
    <HIDEITEM>VIEW_SOURCE</HIDEITEM>
```



```

        <HIDEITEM>HOME_TB</HIDEITEM>
        ...
        <HIDEITEM>RELATIVE_DATE_FILTER_MENU_ITEM</HIDEITEM>
        <HIDEITEM>EDIT_TITLE</HIDEITEM>
    </EXPERIENCE_LEVEL>
</EXPERIENCE_LEVELS>

```

Hide individual features by enclosing them within HIDEITEM tags.

Defining an experience level

Define an experience level by performing the following steps:

- Increment the number of level definitions in the file.
When you add an experience level, you increment this number by one:

```
<NUMBER_OF_LEVELS>3</NUMBER_OF_LEVELS>
```

- Set the default level.
The number of the default experience level for all users. The levels are numbered as their definitions appear in the file, starting from one. The following line selects the second level, Standard, as the default level:

```
<DEFAULT_EXPERIENCE_LEVEL>2</DEFAULT_EXPERIENCE_LEVEL>
```

- Create a new level and specify its name.
A unique alphanumeric string defines the skin name. Actuate Information Console's Options—General page uses the skin name to identify the experience level to Information Console users.

You also can provide multiple display names. Each display name includes an alphanumeric name and the locale for the name. Actuate Analytics Cube Viewer uses the display name for the appropriate locale. The following example shows English and Spanish display names:

```

<DISPLAY_NAME>
    <LOCALE_ID>en_US</LOCALE_ID>
    <NAME>Standard</NAME>
</DISPLAY_NAME>
<DISPLAY_NAME>
    <LOCALE_ID>es_ES</LOCALE_ID>
    <NAME>Estándar</NAME>
</DISPLAY_NAME>

```

- List the Actuate Analytics Cube Viewer features that the level hides.
The Administrator can create and modify experience level definitions to enable or deny access to Actuate Analytics Cube Viewer features. The types of features that you can hide include the following features:

- Horizontal bars, such as the entire banner or toolbar

- Toolbar buttons
- Menu items for toolbar buttons
- Context menu items for table views, bar chart views, and line chart views
- Axis context menu items, such as the context menu that appears when you right-click axis components of a chart

Every experience level definition must include the following entries because these features are not supported in Actuate Analytics:

- `<HIDE_ITEM>ADMIN_LOG_ON</HIDEITEM>`
- `<HIDE_ITEM>EDIT_TITLE</HIDEITEM>`
- `<HIDE_ITEM>VIEW_SOURCE</HIDEITEM>`

Adding an experience level to a functionality level

After you create an experience level, you can add it to a functionality level. Each functionality level includes a list of available experience levels. Every built-in functionality level contains the following lines:

```
<AnalyticsExperienceLevel>Novice</AnalyticsExperienceLevel>
<AnalyticsExperienceLevel>Standard</AnalyticsExperienceLevel>
<AnalyticsExperienceLevel>Advanced</AnalyticsExperienceLevel>
```

You can remove these entries and add others as needed for every functionality level. Every functionality level needs at least one experience level.

Configuring the BIRT Viewer and Interactive Viewer

The BIRT Viewer provides the ability to view a BIRT report. The Interactive Viewer supports modifying many aspects of the report's layout and formatting. These viewers are available in Information Console with the appropriate licensed iServer system option. They are also available as Java Components. Parameters in `web.xml` configure these viewers. For information on those configuration parameters, see *Working with Actuate BIRT Viewers*.

Configuring BIRT Studio

BIRT Studio is a report design tool that you use to design BIRT reports. This designer is available in Information Console with the appropriate licensed iServer system option. It is also available as a Java Component. Parameters in `web.xml` configure it. For information on those configuration parameters, see *Using BIRT Studio - iServer Edition*.

Configuring BIRT Data Analyzer

BIRT Data Analyzer extends the functionality of BIRT Interactive Viewer to perform analytics on a cross tab. You can configure performance enhancements for Data Analyzer in web.xml. For information on those configuration parameters, see *Using BIRT Data Analyzer*.

Actuate Information Console URIs

This chapter contains the following topics:

- Actuate Information Console URIs overview
- Actuate Information Console URIs quick reference
- Common URI parameters
- Information Console Struts actions
- Actuate Information Console URIs reference
- Actuate BIRT Viewer URIs reference
- Actuate Viewer URIs reference

Actuate Information Console URIs overview

This chapter describes Actuate Information Console URIs. Information Console JSPs manage content. The following sections provide quick reference tables and detailed reference information about Actuate Information Console URIs. An Actuate Information Console URI is a directive to Actuate Information Console to perform an action, such as showing a list of files.

Information Console pages use the .do extension for Struts action mapping to a page. The complete page name appears as part of the reference material. Actuate Information Console page and folder names are case sensitive.

Information Console supports two viewers, which have specific URLs associated with them. The detailed reference material for Information Console and the viewers is divided into the following categories:

- Actuate Information Console URIs reference
- Actuate BIRT Viewer URIs reference
- Actuate Viewer URIs reference

Actuate Information Console URIs quick reference

Table 4-1 lists the Actuate Information Console URIs. For more information about the Information Console directory structure, see “Understanding Information Console directory structure” in Chapter 2, “Creating a custom Information Console web application.”

Table 4-1 Actuate Information Console URI pages

Actuate Information Console page	Description
about page	Displays information about Actuate Information Console.
authenticate page	Performs authentication and maintains user, cluster, and volume information.
banner page	Displays a banner at the top of each Actuate Information Console page.
browse file page	Provides file and folder browsing functionality for the submit request pages.
browse page	See browse file page.
calendar page	Provides calendar functionality for submit request’s scheduling feature.

Table 4-1 Actuate Information Console URI pages (continued)

Actuate Information Console page	Description
canceljob page	See request drop page.
channels page	Displays the channels property sheet.
completed request page	Lists all completed requests.
create folder page	Creates a folder.
create query page	Creates a query.
delete file status page	Displays whether a file was successfully deleted.
dashboard page	Provides the dashboard interface.
delete job page	Deletes a scheduled job.
delete status page	Deletes the completed job notice.
detail page	Supports error handling and presenting object details.
do_update page	See options page.
drop page	Supports deleting files or canceling running jobs.
error page	Retrieves an error message from the exception or the request and displays it.
execute page	Runs a query.
execute query page	Submits a run Actuate Query job to the server.
execute report page	Submits a run report job request to the server.
general options page	Displays the general user settings and environment settings property sheet.
getfiledetails page	See file or folder detail page.
getfolderitems page	See file and folder index page.
getjobdetails page	See request detail page.
get saved search page	Executes a saved search.
home page	Provides the link from the My Folder button to the Actuate Information Console home page.
list page	Supports listing channels, channel contents, and Encyclopedia objects.
login banner page	Provides the banner for the Actuate Information Console login page.
login page	Logs into the reporting web application.

(continues)

Table 4-1 Actuate Information Console URI pages (continued)

Actuate Information Console page	Description
logout page	Logs the user out of the current session and clears all user settings, such as filters.
notification page	Supports specifying the current user's request notification options.
options page	Updates options and user settings. See also options index page.
output page	Presents a form to specify output information for report jobs, such as report object name and location.
page not found page	Displays an error message when a JSP is unavailable in Information Console.
parameters page	Presents a list of the request parameters.
pending page	Lists all requests awaiting execution.
ping page	Diagnostics for Actuate BIRTiServer System components.
print page	Prints report documents in PDF format.
privileges page	Sets file and folder privileges.
request search page	Presents the search request form, then performs the search.
running page	Lists all requests currently executing.
save as page	Supports downloading the report document in various output formats.
schedule page	Presents a form for specifying scheduled report job request properties, such as date, time, recurring request, and immediate report job run.
scheduled job page	Lists all requests awaiting execution at specified dates and times.
search folders page	Searches folders recursively for files and folders.
search frame page	Processes report document search criteria.
search report page	Submits report document search criteria, obtains search results, and presents the results to the user.
search toolbar page	Builds and displays the search toolbar.
selectjobs page	See requests index page.
submit job page	Submits a scheduled job request to the server.
submit page	Copies or reruns a completed query.

Table 4-1 Actuate Information Console URI pages (continued)

Actuate Information Console page	Description
viewer page for BIRT reports	Displays BIRT documents and the toolbar.
view cube page	Accesses an Actuate Analytics cube or view.
view default page	Displays the Actuate e.report document in the user's preferred format, set in User Preferences.
view frame set page	Displays the Actuate e.report document along with the navigation bar.
view navigation page	Displays the navigation bar for the Actuate e.report DHTML Viewer.
view TOC page	Displays the Actuate e.report document's table of contents.

Common URI parameters

All Actuate Information Console URIs have the parameters shown in Table 4-2. String values that are too long are truncated for all parameters. The web browser that you use determines the length of parameters. The common URI parameters support Actuate Information Console authentication using cookies.

Table 4-2 Common Actuate Information Console URI parameters

URI parameter	Description
forceLogin	True to force a login, False to display the login page. The default is False. For example, when switching between Encyclopedia volumes and using an Information Console security manager class, set forceLogin=true to force the Information Console Login module to call the security manager to perform the login operation. The login operation is described in "Understanding the authentication process," in Chapter 9, "Using Actuate Information Console security."
iPortalID	The unique authentication ID assigned to the user upon successful login. Use this parameter in conjunction with the userID parameter to ensure that a user's personalized settings appear in the Information Console pages.

(continues)

Table 4-2 Common Actuate Information Console URI parameters (continued)

URI parameter	Description
locale	The current user's locale, such as U.S. English (en-US).
password	The password associated with the userID.
serverURL	The URI that accesses the Actuate BIRT iServer, such as <code>http://Services:8000</code> .
timezone	The current user's time zone.
userID	The user's unique identifier, required to log in to the repository. Use this parameter in conjunction with the iPortalID parameter to ensure that a user's personalized settings appear in the Information Console pages.
volume	The volume to which the user connects. This parameter overrides the volume from the <code>__vp</code> parameter if they are used together.
__vp	The name of a server configured in VolumeProfile.xml. Information Console uses the volume information in a VolumeProfile entry except when a volume parameter specifies a different one.

The following Information Console URI shows most of the common URI parameters in use:

```
http://localhost:8700/iportal/dashboard?folder=/Training
    &volume=Encyc2&locale=en_AU&userID=Mike&password=pw123
    &serverURL=http://server1:8000&timeZone=Australia/Perth
```

This URI lists the contents of the Training folder in the Encyc2 Encyclopedia volume on the Actuate BIRT iServer named server1 at port 8000. The locale is set to Australian English and the time zone is Australia/Perth (GMT plus eight hours). The user is Mike and the password is pw123. The password is shown in plain text, as entered.

If the server and volume information for server1 above is configured as a Volume Profile, you can use a simplified URL as shown in the following lines:

```
http://localhost:8700/iportal/dashboard?folder=/Training
    &__vp=server1&locale=en_AU&userID=Mike&password=pw123
    &timeZone=Australia/Perth
```

Information Console Struts actions

The following tables summarizes the global forwards and actions defined in struts-config.xml.

Table 4-3 lists the global forwards defined in struts-config.xml.

Table 4-3 Actuate Information Console global forwards

Action	Forward
authexpired	/login.do
browsefile	/browsefile.do
canceljob	/canceljob.do
createquery	/query/create.do
da	/da
dashboard	/dashboard
deletefile	/deletefile.do
deletejob	/deletejob.do
deletejobnotice	/deletejobnotice.do
deletejobschedule	/deletejobschedule.do
downloadfile	/servlet/DownloadFile
error	/private/common/errors/errorpage.jsp
erroreportlet	/private/common/errors/errorreportlet.jsp
executedocument	/executedocument.do
executequery	/query/execute.do
executereport	/executereport.do
getjobdetails	/getjobdetails.do
getrequesterjobdetails	/getrequesterjobdetails.do
getsavedsearch	/viewer/getsavedsearch.do
goto	/private/common/goto.jsp
iv	/iv
login	/login.do
logout	/logout.do
requestercanceljob	/requestercanceljob.do
requesterdeletejob	/requesterdeletejob.do
requesterdeletejob schedule	/requesterdeletejobshedule.do
skinerror	/private/common/errors/error.jsp
submitquery	/query/submit.do

(continues)

Table 4-3 Actuate Information Console global forwards (continued)

Action	Forward
viewcube	/viewcube.do
viewframeset	/viewer/viewframeset.jsp
viewpage	/servlet/ViewPage
viewsoi	/viewsoi.do
wr	/wr

Table 4-4 lists the action, input JSP, and forward name and path defined in struts-config.xml.

Table 4-4 Actuate Information Console actions

Action	Input JSP	Forward name path
/analyticsbrowse folder	/iportal/activePortal /private/cubeviewer /analyticsbrowsefolder.jsp	name=success path=/iportal/activePortal/private cubeviewer/analyticsbrowsefolder.jsp
/browsefile	/iportal/activePortal /private/newrequest /browse.jsp	name=success path=/iportal/activePortal/private newrequest/browse.jsp name=browseFile path=/iportal/activePortal/private common/browseFile.jsp
/browseportletfile	/iportal/portlets /browsefile.jsp	name=success path=/iportal/portlets/browsefile.jsp
/canceljob		name=success path=/iportal/activePortal/private jobs/joboperationstatus.jsp
/cancelreport		name=Succeeded path=/iportal/activePortal/viewer closewindow.jsp name=Failed path=/iportal/activePortal/viewer closewindow.jsp?status=failed name=Inactive path=/iportal/activePortal/viewer closewindow.jsp?status=inactive

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/createfolder		name=success path=/getfolderitems.do name=cancel path=/getfolderitems.do name=showform path=/iportal/activePortal/private /filesfolders/createfolder.jsp
/cubedetail		name=success path=/servlet/DownloadFile
/customize		name=success path=/iportal/activePortal/private /customization/skinmanager.jsp name=downloadwar path=/servlet/CacheDownload
/deletefile		name=success path=/iportal/activePortal/private /filesfolders/deletefilestatus.jsp name=error path=/iportal/activePortal/private /filesfolders/deletefilestatus.jsp name=confirm path=/iportal/activePortal/private /filesfolders/confirm.jsp
/deletejob		name=success path=/iportal/activePortal/private /jobs/joboperationstatus.jsp
/deletejobnotice		name=success path=/iportal/activePortal/private /jobs/joboperationstatus.jsp
/deletejobschedule		name=success path=/iportal/activePortal/private /jobs/joboperationstatus.jsp
/editTableRow	/iportal/activePortal /private/parameters /table/roweditor.jsp	name=close path=/iportal/activePortal/private /parameters/table/close.jsp name=tableRowEditor path=/iportal/activePortal/private /parameters/table/roweditor.jsp

(continues)

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/executedocument		name=success path=/executereport.do
/executereport	/private/newrequest /newrequest.jsp	name=viewbirt path=/iv name=viewreport path=/servlet/DownloadFile name=viewroi path=/viewer/viewframeset.jsp name=viewessreport path=/servlet name=wait path=/iportal/activePortal/private /newrequest/waitforexecution.jsp
/filefoldersprivilege	/iportal/activePortal /private/filesfolders /privilege.jsp	name=success path=/getfolderitems.do
/getfiledetails		name=success path=/iportal/activePortal/private /filesfolders/filedetail.jsp
/getfolderitems		name=success path=/iportal/activePortal/private /filesfolders/filefolderlist.jsp name=dashboard path=/dashboard
/getjobdetails		name=success path=/iportal/activePortal/private /jobs/getjobdetails.jsp
/getnoticejobdetails		name=success path=/iportal/activePortal/private /jobs/getjobdetails.jsp
/getportletfolder items		name=success path=/iportal/portlets/filefolderlist /filefolderlistportlet.jsp
/getrequesterjob details		name=success path=/iportal/activePortal/private /jobs/getrequesterjobdetails.jsp

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/iPortalLogin	/iportal/login.jsp	name=landing path=/landing.jsp name=iPortalLoginForm path=/login.jsp
/iv	/iportal/activePortal /private/newrequest /newrequest.jsp	name=iv path=/iv name=viewbirt path=/iv
/login	/iportal/activePortal /private/login.jsp	name=loginform path=/iportal/activePortal/private /login.jsp name=success path=/getfolderitems.do name=dashboard path=/dashboard name=ajclogin path=/ajclanding.jsp name=landing path=/landing.jsp
/logout		name=landing path=/landing.jsp
/options	/iportal/activePortal /private/options /options.jsp	name=success path=/iportal/activePortal/private /options/options.jsp name=saved path=/getfolderitems.do name=dashboard path=/iportal/activePortal/private /options/options.jsp
/options/save	/iportal/activePortal /private/options /options.jsp	name=success path=/getfolderitems.do name=saved path=/getfolderitems.do
/ping		name=success path=/iportal/activePortal/private /diagnosis/pingresponse.jsp

(continues)

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/query/create	/iportal/activePortal /private/query /create.jsp	name=success path=/iportal/activePortal/private /query/create.jsp name=cancel path=/getfolderitems.do name=overwritePrompt path=/iportal/activePortal/private /query/fileexists.jsp name=confirmation path=/iportal/activePortal/private /query/confirmation.jsp
/query/execute	/iportal/activePortal /private/query /execute.jsp	name=success path=/iportal/activePortal/private /query/execute.jsp name=cancel path=/getfolderitems.do
/query/submit	/iportal/activePortal /private/query /execute.jsp	name=success path=/iportal/activePortal/private /query/execute.jsp name=cancel path=/getfolderitems.do name=confirmation path=/iportal/activePortal/private /query/confirmation.jsp
/requestercanceljob		name=success path=/iportal/activePortal/private /jobs/requesterjoboperationstatus.jsp
/requesterdeletejob		name=success path=/iportal/activePortal/private /jobs/requesterjoboperationstatus.jsp
/requesterdeletejob schedule		name=success path=/iportal/activePortal/private /jobs/requesterjoboperationstatus.jsp
/searchfiles		name=success path="/iportal/activePortal/private /filesfolders/search/filefolderlist.jsp
/selectchannels		name=channellist path=/iportal/activePortal/private /channels/channellist.jsp

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/selectjobnotices		name=success path=/iportal/activePortal/private/channels/channelnoticelist.jsp
/selectjobs		name=success path=/iportal/activePortal/private/jobs/selectjobs.jsp
/searchfiles		name=success path=/iportal/activePortal/private/filesfolders/search/filefolderlist.jsp
/skinedit	/customize.do	name=success path=/iportal/activePortal/private/customization/skinedit.jsp
/submitjob	/iportal/activePortal/private/newrequest/newrequest.jsp	name=createquery path=/query/create.do name=query path=/query/submit.do name=success path=/iportal/activePortal/private/newrequest/submitjobstatus.jsp name=viewreport path=/servlet/DownloadFile name=viewroi path=/iportal/activePortal/viewer/viewframeset.jsp name=viewessreport path=/servlet
/subscribeChannel	/iportal/activePortal/private/channels/channelsubscribe.jsp	name=success path=/selectchannels.do
/tableList	/iportal/activePortal/private/parameters/table/tableparameters.jsp	name=close path=/iportal/activePortal/private/parameters/table/close.jsp name=tableParamList path=/iportal/activePortal/private/parameters/table/tableparameters.jsp
/treebrowser		name=success path=/iportal/activePortal/private/filesfolders/treebrowser.jsp

(continues)

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/updatechannel		name=success path=/iportal/activePortal/private/channels/channeloperationstatus.jsp
/uploadimage		name=success path=/iportal/activePortal/private/customization/fileupload.jsp
/uploadlicense		name=success path=/iportal/admin/fileupload.js
/viewcube		name=analyticsbrowsefolder path=analyticsbrowsefolder.do name=analyticsexplevel path=servlet/AnalyticsExpLevel name=downloadcube path=servlet/DownloadFile name=dbdetail path=cubedetail.do name=dbstore path=servlet/CubeStore name=odbotunnel path=servlet/OdboTunnel name=success path=/iportal/activePortal/private/cubeviewer/viewcube.jsp
/viewer		name=success
/getsavedsearch		path=/getfolderitems.do name=searchreport path=/iportal/activePortal/viewer/searchreportpage.jsp name=requestsearch path=/iportal/activePortal/viewer/requestsearch.jsp
/viewer/savesearch	/iportal/activePortal/viewer/savesearch.jsp	name=success path=/iportal/activePortal/viewer/requestsearch.jsp name=browse path=/browsefile.do
/viewsoi	/iportal/activePortal/private/newrequest/newrequest.jsp	name=viewessreport path=/servlet

Table 4-4 Actuate Information Console actions (continued)

Action	Input JSP	Forward name path
/waitforreport execution	/iportal/activePortal /private/newrequest /waitforexecution.jsp	name=success path=/iportal/activePortal/viewer /viewreport.jsp name=fail path=/iportal/activePortal/viewer /closewindow.jsp

Actuate Information Console URIs reference

This section provides the detailed reference for Actuate Information Console URIs. In the definitions, <context root> represents the name of your Actuate Information Console context root, initially iportal. Table 4-5 lists the topics this chapter covers and the file names discussed in each topic. All pages are under the Information Console context root.

Table 4-5 Actuate Information Console pages

Topic	Information Console file
about page	iportal\activePortal\private\options\about.jsp
authenticate page	iportal\activePortal\authenticate.jsp
banner page	iportal\activePortal\private\common\banner.jsp
browse file page	browsefile.do iportal\activePortal\private\query\browse.jsp
calendar page	iportal\activePortal\private\newrequest \calendar.jsp
channels page	iportal\activePortal\private\options\channels.js
completed request page	iportal\activePortal\private\jobs\completedjob.jsp
create folder page	createfolder.do
create query page	create.do iportal\activePortal\private\query\create.jsp
dashboard page	DashboardServlet
delete file status page	iportal\activePortal\private\filesfolders \deletefilestatus.jsp
delete job page	deletejob.do

(continues)

Table 4-5 Actuate Information Console pages (continued)

Topic	Information Console file
delete status page	deletejobnotice.do
detail page	
■ error detail page	iportal\activePortal\errors\detail.jsp getfiledetails.do
■ file or folder detail page	iportal\activePortal\private\filesfolders\filedetail.jsp
■ request detail page	getjobdetails.do iportal\activePortal\private\jobs\getjobdetails.jsp
drop page	
■ file or folder drop page	deletefile.do
■ request drop page	canceljob.do
error page	errors\error.jsp iportal\activePortal\private\common\errors\error.jsp
execute page	query\execute.do
execute query page	iportal\activePortal\query\execute.jsp
execute report page	executereport.do
general options page	iportal\activePortal\private\options\general.jsp
get saved search page	iportal\activePortal\viewer\getsavedsearch.do
home page	iportal\activePortal\private\common\breadcrumb.jsp
index page	
■ file and folder index page	getfolderitems.do iportal\activePortal\private\filesfolders\filefolderlist.jsp
■ new request index page	executereport.do
■ options index page	options.do iportal\activePortal\private\options\options.jsp
■ requests index page	selectjobs.do iportal\activePortal\private\jobs\selectjobs.jsp

Table 4-5 Actuate Information Console pages (continued)

Topic	Information Console file
list pages	
■ channels list page	selectchannels.do iportal\activePortal\private\channels\channellist.jsp
■ channel contents list page	iportal\activePortal\private\channels\channelnoticelist.jsp
■ file and folder list page	getfolderitems.do iportal\activePortal\private\filesfolders\filefolderlist.jsp
login banner page	iportal\activePortal\private\login_banner.jsp
login page	login.do iportal\activePortal\private\login.jsp
logout page	logout.do
notification page	iportal\activePortal\private\options\notification.jsp
options page	options.do iportal\activePortal\private\options\options.jsp
output page	iportal\activePortal\private\newrequest\output.jsp
page not found page	iportal\activePortal\errors\pagenotfound.jsp
parameters page	iportal\activePortal\private\newrequest\parameters.jsp
pending page	iportal\activePortal\private\jobs\pendingjob.jsp
ping page	ping.do iportal\activePortal\private\diagnosis\pingresponse.jsp
print page	iportal\activePortal\viewer\print.jsp
running page	iportal\activePortal\private\jobs\runningjob.jsp
save as page	iportal\activePortal\viewer\saveas.jsp
schedule page	iportal\activePortal\private\newrequest\schedule.jsp
scheduled job page	iportal\activePortal\private\jobs\scheduledjob.jsp
search folders page	searchfiles.do iportal\activePortal\private\filesfolders\search\filefolderlist.jsp

(continues)

Table 4-5 Actuate Information Console pages (continued)

Topic	Information Console file
submit job page	submitjob.do iportal\activePortal\private\newrequest \submitjobstatus.jsp
submit page	query\submit.do iportal\activePortal\private\query\execute.jsp
viewer page for BIRT reports	IVServlet
view cube page	viewcube.do

about page

Displays the About page, containing information about Actuate Information Console. Called when the user chooses the About tab on the Options page.

The default about page for Information Console is similar to Figure 4-1.



Figure 4-1 Information Console about page

Name <context root>\iportal\activePortal\private\options\about.jsp

Parameters The about page uses the common URI parameters.

Used by iportal\activePortal\private\options\optionspage.jsp

authenticate page

Performs user authentication and maintains the user, cluster, and volume information authentication data during the user's session. Pages that require validation of user credentials before permitting access to folders or files use the authenticate page. In Information Console, only pages for the DHTML Viewer

use the authenticate page. The remaining Information Console pages use the Struts framework for authentication.

Name	<context root>iportal\activePortal\authenticate.jsp
Parameters	The authenticate page uses the common URI parameters.
Used by	iportal\activePortal\errors\error.jsp iportal\activePortal\viewer\closewindow.jsp iportal\activePortal\viewer\print.jsp iportal\activePortal\viewer\requestsearch.jsp iportal\activePortal\viewer\saveas.jsp iportal\activePortal\viewer\searchframe.jsp iportal\activePortal\viewer\searchreport.jsp iportal\activePortal\viewer\searchtoolbar.jsp iportal\activePortal\viewer\viewdefault.jsp iportal\activePortal\viewer\viewframeset.jsp iportal\activePortal\viewer\viewnavigation.jsp iportal\activePortal\viewer\viewreport.jsp iportal\activePortal\viewer\viewtoc.jsp iportal\activePortal\private\newrequest\waitforexecution.jsp

banner page

Provides the banner that appears across the top of all Actuate Information Console web pages. The default banner displays the Actuate logo, user name, and license, and provides links for Logout, Options, and Help. The banner page obtains the user name from variables maintained by the authenticate page.

Name	<context root>iportal\activePortal\private\common\banner.jsp
Used by	iportal\activePortal\private\login.jsp iportal\activePortal\private\channels\channelnoticelist.jsp iportal\activePortal\private\channels\channeloperationstatus.jsp iportal\activePortal\private\filesfolders\deletefilestatus.jsp iportal\activePortal\private\filesfolders\filedetail.jsp iportal\activePortal\private\filesfolders\filefolderlist.jsp iportal\activePortal\private\jobs\getjobdetails.jsp iportal\activePortal\private\jobs\joboperationstatus.jsp iportal\activePortal\private\jobs\selectjobs.jsp iportal\activePortal\private\newrequest\newrequest.jsp iportal\activePortal\private\newrequest\newrequest2.jsp iportal\activePortal\private\newrequest\submitjobstatus.jsp iportal\activePortal\private\options\options.jsp iportal\activePortal\private\query\create.jsp iportal\activePortal\private\query\execute.jsp

browse file page

Contains file and folder browsing functionality used by submit request pages.

Name <context root>\browsefile.do

<context root>\portal\activePortal\private\query\browse.jsp

Parameters workingFolder is the name of the folder for which to display contents in the folder browser window. The browse file page also uses the common URI parameters.

Used by portal\activePortal\private\newrequest\browse.jsp
portal\activePortal\private\query\browse.jsp

calendar page

Provides calendar functionality for the submit request scheduling feature.

Name <context root>\portal\activePortal\private\newrequest\calendar.jsp

Used by portal\activePortal\private\newrequest\newrequestpage.jsp
portal\activePortal\private\query\createpage.jsp
portal\activePortal\private\query\runpage.jsp

channels page

Displays the channels property sheet. The channels page presents a list of all channels available on the current volume. Channels to which the user subscribes appear with their check boxes selected.

The channels page looks like Figure 4-2.

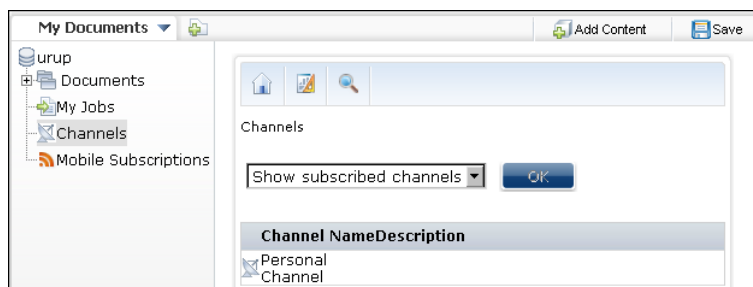


Figure 4-2 Channels page

Users choose which channels they want to see in the list by specifying a filter. For example, to see all Marketing Communications channels, the user might type the filter Mar* in the Filter field. Channels uses the HTTP session variable AcChannelFilter to save the current filter value. AcChannelFilter works if cookies are enabled. For more information, see *Managing an Encyclopedia Volume*.

Name <context root>\portal\activePortal\private\options\channels.jsp
Used by portal\activePortal\private\options\optionspage.jsp

completed request page

Lists all completed requests. The completed request page lists all report jobs that have executed and are available or whose execution failed.

The completed request page looks like Figure 4-3.

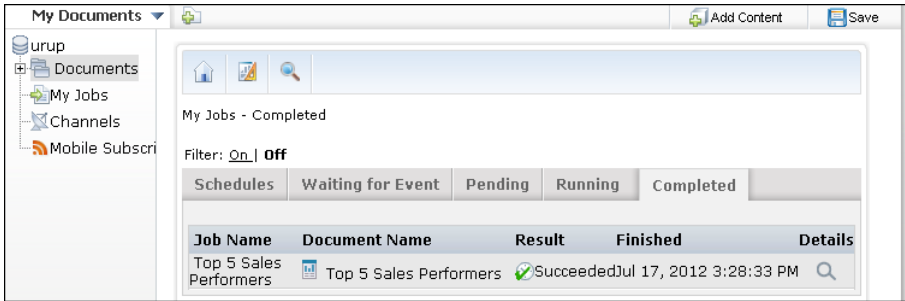


Figure 4-3 Completed request page

Name <context root>\portal\activePortal\private\jobs\completedjob.jsp
Parameters The completed request page uses the common URI parameters.
Used by portal\activePortal\private\jobs\selectjobscontent.jsp

create folder page

Creates a folder in the current Encyclopedia volume. Createfolder.do uses <context root>\portal\activePortal\private\filesfolders\createfolder.jsp to create the new folder.

Name <context root>\createfolder.do

Parameters Table 4-6 lists and describes the parameters for the create folder page. The create folder page also uses the common URI parameters.

Table 4-6 Parameters for create folder URI

URI parameter	Description
workingFolderID	The ID of the folder to contain the new folder. Specify either workingFolderID or workingFolderName.
workingFolderName	The name of the folder to contain the new folder. Specify either workingFolderID or workingFolderName.

Used by Not applicable.

create query page

Creates a query from a data object executable (.dox) file using the Actuate Query Wizard. create.do uses the HTML code in <context root>\iportal\activePortal\private\query\createpage.jsp to display the query data.

Name <context root>\query\create.do
<context root>\iportal\activePortal\private\query\create.jsp

Parameters Table 4-7 describes the parameter for the create query page. The create query page also uses the common URI parameters.

Table 4-7 Parameter for create query URI

URI parameter	Description
__executableName	The full path name of the DOV or DOX to use for the query

Used by Not applicable.

dashboard page

Provides the dashboard interface servlet for information console.

Name <context root>\dashboard

Used by Not applicable.

delete file status page

Summarizes the result of a deletion performed by the drop page and indicates whether a file was successfully deleted. The delete file status page includes authenticate to obtain user session data. Information Console performs the deletion as part of an action and then forwards to the delete file status page.

Name <context root>\portal\activePortal\private\filesfolders\deletefilestatus.jsp
Used by Not applicable.

delete job page

Deletes the specified job, then redirects the page to a completion status page. Specify the name or the ID of the job to delete.

The default redirection JSP is <context root>\portal\activePortal\private\jobs\joboperationstatus.jsp.

Name <context root>\deletejob.do
Parameters Table 4-8 lists and describes the parameters for the delete job page. The delete job page also uses the common URI parameters.

Table 4-8 Parameters for delete job URI

URI parameter	Description
jobID	Unique request identifier.
jobName	The name of the job to delete. This parameter is ignored if jobID is also specified.
jobState	The state of the job to delete.
redirect	URI to which to redirect the job deletion page.

Used by Not applicable.

delete status page

Deletes a job notice corresponding to a request. Specify the job notice by name or by ID.

Name <context root>\deletejobnotice.do
The default redirection action forwards to <context root>\portal\activePortal\private\jobs\joboperationstatus.jsp.

Parameters Table 4-9 lists and describes the parameters for the delete status page. The delete status page also uses the common URI parameters.

Table 4-9 Parameters for delete status URI

URI parameter	Description
channelID	The unique identifier of the channel to delete the job notice from.
channelName	The name of the channel to delete the job notice from.
jobID	Unique request identifier.
jobName	The name of the job notice to delete. This parameter is ignored if jobID is also specified.
jobState	The state of the job to delete.
redirect	URL to which to redirect the delete status page.
userName	The name of the user to notify about the deleted job.

Used by Not applicable.

detail page

Displays detailed information about Encyclopedia volume objects. There are three detail pages:

<context root>\iportal\activePortal\errors
<context root>\iportal\activePortal\filesfolders
<context root>\iportal\activePortal\requests

error detail page

Provides a template error page that can be embedded in another page. For example, the view navigation page and the view TOC page use the error detail page to build error pages.

Name <context root>\iportal\activePortal\errors\detail.jsp

Used by iportal\activePortal\private\common\errors\error.jsp
iportal\activePortal\viewer\print.jsp
iportal\activePortal\viewer\saveas.jsp
iportal\activePortal\viewer\searchframe.jsp
iportal\activePortal\viewer\viewdefault.jsp
iportal\activePortal\viewer\viewtoc.jsp

file or folder detail page

Displays detailed information about the selected viewable folder or file. Users request file or folder details by choosing the magnifying glass icon to the right of files or folders listed on the Encyclopedia folder page or breadcrumb. Users can request another document or delete the current file or folder from the file or folder detail page. `filedetail.jsp` uses the HTML code in `<context root>\iportal\activePortal\private\filesfolders\filedetailcontent.jsp` to display the information.

Name `<context root>\getfiledetails.do`
`<context root>\iportal\activePortal\private\filesfolders\filedetail.jsp`

Parameters Table 4-10 lists and describes the parameters for the file or folder detail page. The file or folder detail page also uses the common URI parameters.

Table 4-10 Parameters for file or folder detail URI

URI parameter	Description
name	The full path name of the Encyclopedia object for which to show details, if <code>objectID</code> is not specified.
objectID	The Encyclopedia object’s unique identifier.
version	The Encyclopedia object’s version number. The default is the latest version.

Used by Not applicable.

request detail page

Lists detailed request information for a specified job, as shown in Figure 4-4. `getjobdetails.jsp` uses the HTML code in `<context root>\iportal\activePortal\private\jobs\getjobdetailscontent.jsp` to display the information.

Name `<context root>\getjobdetails.do`
`<context root>\iportal\activePortal\private\jobs\getjobdetails.jsp`

Parameters The request detail page uses the common URI parameters, as shown in Table 4-11.

Table 4-11 Parameters for request detail URI

URI parameter	Description
jobID	The job’s unique identifier
userName	The user that submitted the job
channelName	The channel to receive the request

Used by iportal\activePortal\private\jobs\completedjob.jsp
 iportal\activePortal\private\jobs\pendingjob.jsp
 iportal\activePortal\private\jobs\runningjob.jsp
 iportal\activePortal\private\jobs\scheduledjob.jsp

Job Status

Open

Delete Job

Schedule (Succeeded)

Job Name:

Top 5 Sales Performers

Owner:

Administrator

Priority:

1000

Submitted:

Mar 6, 2012 9:21:23 AM

Started:

Mar 6, 2012 9:21:23 AM

Finished:

Mar 6, 2012 9:22:13 AM

Run Job:

The job was scheduled immediately.

Event Name:

Event Type:


No event

Event Parameter:

Event Status:

Report (Executable)

Name:

 Top 5 Sales Performers

Type:

Actuate BIRT Design

Version number:

1

Version name:

Folder:

/Public/BIRT and BIRT Studio Examples

Output Document

Name:

 Top 5 Sales Performers

Type:

Actuate BIRT Document

Version name:

Headline:

Folder:

/Home/administrator

Figure 4-4 Request detail page

drop page

Deletes one or more files or folders, or cancels a running job.

file or folder drop page

Deletes the specified file or folder. The file or folder drop page includes the authenticate page to obtain user session data.

Name <context root>\deletefile.do

Parameters Table 4-12 lists and describes the parameters for the file or folder drop page. The file or folder drop page also uses the common URI parameters.

Table 4-12 Parameters for file or folder drop URI

URI parameter	Description
ID	The unique identifier of the object to delete.
name	The full path name of the Encyclopedia object to delete. Multiple name parameters, to delete more than one file or folder at a time, are allowed. This parameter is ignored if ID is also specified.
redirect	URI to navigate to upon completion. The default redirect page is processedaction_status.

Used by Not applicable.

request drop page

Cancels a running job.

Name <context root>\canceljob.do

Parameters Table 4-13 lists and describes the parameters for the request drop page. The request drop page also uses the common URI parameters.

Table 4-13 Parameters for request drop URI

URI parameter	Description
jobID	The unique identifier of the Encyclopedia object to delete.
jobName	The full path name of the Encyclopedia object to delete. This parameter is ignored if jobID is also specified.
jobState	The state of the job to delete. processedaction_status uses jobState to build a link to pass to the list of scheduled and completed jobs.
redirect	URI to navigate to upon completion. The default redirect page is processedaction_status.

Used by Not applicable.

error page

Displays the specified error message. Information Console uses two pages. The DHTML Viewer uses <context root>\iportal\activePortal\errors\error.jsp. All other Information Console code uses <context root>\iportal\activePortal\private\common\errors\error.jsp.

Name <context root>iportal\activePortal\errors\error.jsp
<context root>iportal\activePortal\private\common\errors\error.jsp

Used by iportal\activePortal\private\login.jsp
iportal\activePortal\private\common\closewindow.jsp
iportal\activePortal\private\common\sidebar.jsp
iportal\activePortal\private\common\errors\errorpage.jsp
iportal\activePortal\private\options\options.jsp
iportal\activePortal\private\query\create.jsp
iportal\activePortal\private\query\execute.jsp
iportal\activePortal\private\templates\template.jsp
iportal\activePortal\viewer\closewindow.jsp
iportal\activePortal\viewer\print.jsp
iportal\activePortal\viewer\saveas.jsp
iportal\activePortal\viewer\searchframe.jsp
iportal\activePortal\viewer\searchreport.jsp
iportal\activePortal\viewer\viewframeset.jsp

execute page

Executes a query from a data object executable (.dox) file using the Actuate Query Wizard. When executing a report job or query, a Cancel button appears after a specified wait time passes. Change the time by setting the EXECUTE_REPORT_WAIT_TIME configuration parameter in the appropriate Actuate Information Console configuration file.

Name <context root>\query\execute.do

Parameters Table 4-14 describes the parameter for the execute page. The execute page also uses the common URI parameters.

Table 4-14 Parameter for execute URI

URI parameter	Description
__executableName	The full path name of the .DOV or .DOX file to use for the query

Used by Not applicable.

execute query page

Executes an Actuate query on the Actuate BIRT iServer.

Name <context root>\portal\activePortal\private\query\execute.jsp

Parameters Table 4-15 lists and describes the parameters for the execute query page.

Table 4-15 Parameters for execute query URI

URI parameter	Description
__ageDays	Use with __ageHours to determine how long output objects exist before they are deleted. Use only if __archivePolicy is set to age. __ageDays can be any positive number.
__ageHours	Use with __ageDays to determine how long output objects exist before they are deleted. Use only if __archivePolicy is set to age. __ageHours can be any positive number.
__archiveBefore Delete	Indicate whether to archive the output objects of the current request before deleting them, according to __archivePolicy's setting. Set this parameter to True to archive objects before deleting them. The default value is False. This parameter has no effect if __archivePolicy is set to folder.
__ifExists	Indicates whether to overwrite an existing or create a new file, up to an optional limit. Values are: <ul style="list-style-type: none">■ create — creates a new output file.■ create[n] — creates a new output file up to n versions. For example, to create no more than seven versions, use create7.■ replace — overwrite any existing output.
__jobName	The name for the job to submit.
__outputName	Specifies a name for the query output.
postback	Set to True to execute the job immediately, False to populate the page with parameters.

(continues)

Table 4-15 Parameters for execute query URI (continued)

URI parameter	Description
__archivePolicy	The archive policy to implement for the objects created as output for the current request. Values are folder, age, and date. Set this parameter to folder to use the archive policy that is already set for the folders to which the output is distributed. Set this parameter to age to delete objects older than a specific time period. Set this parameter to date to delete objects on a specific date.
__dateToDelete	The date on which to delete the output objects of the current request. Use only if __archivePolicy is set to date. __dateToDelete must be a date in a locale-specific format. The default format is mm/dd/yyyy.
__executableName	The name of the executable file for this request.
__priority	Specifies the job submission priority. Values are a number from 1 to 1000, High (800), Medium (500), and Low (200). Do not use with __priorityValue.
__priorityValue	Specifies a number corresponding to the job submission priority. Do not use with __priority.
__progressive	Indicates whether to display the report document after it generates. If False, the report document displays after it generates. If True, the report document displays progressively, as it generates. Applies only to run report jobs.
__timeToDelete	Specifies a time at which to delete an archived report document. Applies only to scheduled report jobs.
__versionName	Contains a string value for the new version name of the job's report document output. The value can include a date/time expression enclosed in braces, {}, to ensure a unique version name.
__viewFormat	Contains a string value specifying the output format for the query. Values are XLS, PDF, e.Analysis, and DHTML, the default.

Used by Not applicable.

execute report page

Submits a run report job request to the Actuate BIRT iServer. When executing a report job or query, a Cancel button appears after a specified wait time passes.

Change the time by setting the EXECUTE_REPORT_WAIT_TIME configuration parameter in the appropriate Actuate Information Console configuration file. For reports that accept run-time parameters, you can set the parameter in the URL by adding an ampersand (&), the parameter name, and an equal(=) sign, followed by the parameter value in quotes.

Name <context root>\executereport.do

Parameters Table 4-16 lists and describes the parameters for the execute report page. The execute report page also uses the common URI parameters.

Table 4-16 Parameters for execute report URI

URI parameter	Description
__ageDays	Use with __ageHours to determine how long output objects exist before they are automatically deleted. Use only if __archivePolicy is set to Age. __ageDays can be any positive number.
__ageHours	Use with __ageDays to determine how long output objects exist before they are automatically deleted. Use only if __archivePolicy is set to Age. __ageHours can be any positive number.
__archiveBeforeDelete	Indicate whether to archive the output objects of the current request before deleting them, according to __archivePolicy's setting. Set this parameter to True to archive objects before deleting them. The default value is False. This parameter has no effect if __archivePolicy is set to Folder.
__archivePolicy	The archive policy to implement for the objects created as output for the current request. Values are folder, age, and date. Set to folder to use the archive policy that is already set for the folders to which the output is distributed. Set to age to delete objects older than a specific time period. Set to date to delete objects on a specific date.
category_ <categoryname>	Indicates whether to use the <category> dimension, or column, of a data cube. Set to True to include the dimension or False to exclude the dimension. For example, &CATEGORY_status=False, where status is the name of the dimension in the cube.
__dateToDelete	The date on which to delete the output objects of the current request. Use only if __archivePolicy is set to Date. Set __dateToDelete to a date in a locale-specific format. The default format is mm/dd/yyyy.

(continues)

Table 4-16 Parameters for execute report URI (continued)

URI parameter	Description
__executableName	The name of the executable file for this request.
__headline	A descriptive tag line for a report document. Appears on Channel Contents. Use the character string %20 to represent a space in the headline string.
invokeSubmit	Controls whether the browser is redirected to the parameter screen or whether the report job is run immediately. If True, the report job is executed without displaying the parameters. If False, the parameters are displayed. False is the default.
__isnull	Sets the value of the named parameter to null. Use a parameter name as input.
__jobName	The name of the job to execute.
__limit	Indicate whether to limit the number of versions of the output files for the current request. Set __limit to Limit to limit the number of versions. Any other value means that the number of versions is unlimited.
__limitNumber	The number of versions to which to limit the output files for the current request. Use only if __limit is set to Limit. __limitNumber can be any positive number.
measure_ <measurename>	Indicates whether to use the data of a numeric <measurename> measure of a data cube. Set to True to include the measure or False to exclude the measure. For example, &MEASURE_size=False, where size is the name of the measure in the cube.
__outputFolderType	Specifies the root of the output file name. Set to Absolute to use the full __outputName value starting from the Encyclopedia volume's root. Set to Personal to use the __outputName value relative to the user's home folder.
__outputDocName	Specifies a name for the output file.
__overwrite	New to create a new version of this report document, or Old to overwrite an existing report document. New is the default.
__priority	Specifies the job submission priority. Values are High, Medium, and Low.
__priorityValue	Specifies a number ranging from 1 to 1000 and corresponding to the job submission priority. Only specify values allowed by your functionality level.

Table 4-16 Parameters for execute report URI (continued)

URI parameter	Description
__progressive	Indicates whether to display the report document after it generates. If False, the report document displays after it generates. If True, the report document displays progressively, as it generates.
__recurringDay	Specifies the scheduled recurring day on which to run the report job. Applies only to scheduled report jobs.
__saveOutput	Indicates whether to write the output document to the Encyclopedia volume. True saves the output in the Encyclopedia volume, applying the document archiving and file creation parameters. False does not save the output.
__serverURL	Contains the URI that accesses the JSP engine, such as http://<iserver machine name>:8700.
__timeToDelete	Specifies a time at which to delete an archived report document. Applies only scheduled report jobs.
__users	Contains the name of the user to notify of this scheduled request. You can notify more than one user. This parameter is valid only for scheduled jobs.
__versionName	Contains a string value for the new version name of this report document. The value can include a date/time expression enclosed in braces, {}, to ensure a unique version name.
__volume	Contains a string value specifying the volume for this report.
__wait	If "wait", Information Console waits for the report generation to be completed before displaying it. If "nowait", Information Console displays the first page right away even if the report job is not completed.

For example, the following URL executes the Sales By Territory.rptdesign report immediately with the Territory run-time parameter set to EMEA:

```
http://localhost:8700/portal/executereport.do?
__requesttype=immediate&__executableName=%2fPublic%2fBIRT and
BIRT Studio Examples%2fSales by Territory.rptdesign&
userid=Administrator&__saveOutput=false&Territory="EMEA"&
invokeSubmit=True
```

Set string parameters to an empty string by adding the parameter to the executereport.do URI with no value following the equal (=) sign.

For example, the following line sets parameterA and parameterB to empty strings:

```
&parameterA=&ParameterB=
```

The following parameter names are reserved for internal use only by the execute report page:

- doframe
- inputfile
- jobType
- name
- selectTab

Used by Not applicable.

general options page

Displays the general user settings and environment settings property sheet for the current user. There are two types of settings:

- User settings that apply only to this user:
 - Change password
 - Change e-mail address
- Environment settings that apply for all browsers on a single local machine:
 - Choose a skin to provide colors, fonts, images, and layout in the graphical user interface (GUI).
 - Choose a view to select a layout for the content area of pages providing lists of files and folders.
 - Set an experience level for Actuate Analytics to enable or disable Cube Viewer features.
 - Enable and disable filter fields for Files and Folders, Channels, and Requests.
 - View documents in the current browser window or in a new browser window.

The general options page appears when the user chooses Options in the Actuate Information Console banner.

Name <context root>\portal\activePortal\private\options\general.jsp

Used by iportal\activePortal\private\options\optionspage.jsp

get saved search page

Executes the specified saved search. The search results appear in the viewer. You need the Read privilege for the saved search file and the Read privilege for the file's corresponding report document.

- Name**

<context root>\viewer\getsavedsearch.do
- Parameters**

Table 4-17 lists and describes the parameters for the get saved search page. The get saved search page also uses the common URI parameters.

Table 4-17 Parameters for get saved search URI

URI parameter	Description
RosFileName	Name of the saved Smart Search ROS file. Specify either RosFileName or RosFileID.
RosFileID	ID of the saved SmartSearch ROS file. Specify either RosFileName or RosFileID.
BasedOnFileID	ID of the ROI or DOI file for the Smart Search criteria to search. The default is the latest version of the file.

home page

Provides two sets of links. On the right side it provides a graphical and a text shortcut link from the My Folder button to the current user's Actuate Information Console home folder. If the Information Console installation supports BIRT Studio, there is another shortcut link, BIRT Studio, to the BIRT Studio. On the left side, it provides the links and other text for the breadcrumb, or path from the repository root to the current folder.

Users access their home page by choosing the My Folder link below the Actuate Information Console page banner. Figure 4-5 shows the default My Folder and breadcrumb links.

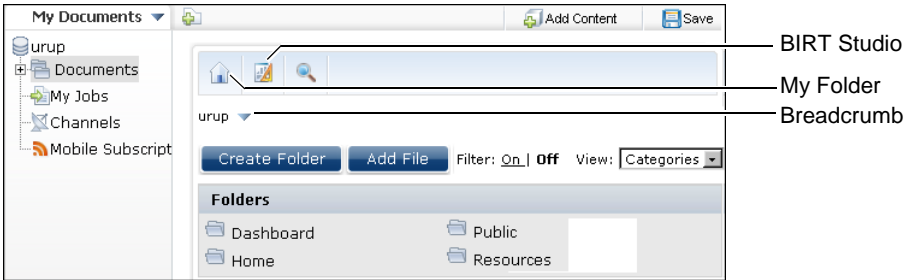


Figure 4-5 My Folder and breadcrumb links

Name <context root>\iportal\activePortal\private\common\breadcrumb.jsp

Used by iportal\activePortal\private\skins\tabbed\templates\mypagetabletemplate.jsp
iportal\activePortal\private\skins\tabbed\templates\template.jsp
iportal\activePortal\private\skins\classic\templates\template.jsp
iportal\activePortal\private\skins\treeview\templates\template.jsp

index page

Provides the entry point and structure for the parts of Actuate Information Console generated from multiple files.

file and folder index page

The default entry point to the Actuate Information Console web application. The file and folder index page provides the entry point and structure to support the Files and Folders functionality. The structure is a table that Actuate Information Console uses to format and present files and folders data. Page content varies depending on the Actuate Information Console directive.

The file and folder index page uses the banner page to provide the reporting web page banner. filefolderlist.jsp uses the HTML code in <context root>\iportal\activePortal\private\filesfolders\filefolderlistcontent.jsp to display files and folders data.

Name <context root>\getfolderitems.do

<context root>\iportal\activePortal\private\filesfolders\filefolderlist.jsp

Parameters Table 4-18 lists and describes the parameters for file and folder index page. The file and folder index page also uses the common URI parameters.

Table 4-18 Parameters for file and folder index URI

URI parameter	Description
startUpMessage	Specifies a message to appear when Actuate Information Console calls this page.
subpage	Specifies the content of the page. Possible values are: <ul style="list-style-type: none"> ■ _list: include list ■ _detail: include detail Specifying any other value for subpage invokes the page not found page.

new request index page

Provides the entry point and structure to support the submit job functionality.

- Name**
- <context root>\executereport.do
- Parameters**
- Table 4-19 describes the parameter for the new request index page. The new request index page also uses the common URI parameters.

Table 4-19 Parameter for new request index URI

URI parameter	Description
homeFolder	The location of the My Documents folder

options index page

Provides the entry point and structure to support the Options functionality. The structure is a table that Actuate Information Console uses to format and present files and folders data. The default table includes the banner across the top of the page, the side menu on the left side of the page, and a container for page content. Page content varies depending upon the Actuate Information Console directive.

The options index page uses the banner page to provide the reporting web page banner. options.jsp uses the HTML code in <context root>\iportal\activePortal\private\options\optionspage.jsp to display the options data.

- Name**
- <context root>\options.do

<context root>\iportal\activePortal\private\options\options.jsp
- Parameters**
- Table 4-20 describes the parameter for the options index page. The options index page also uses the common URI parameters.

Table 4-20 Parameter for options index URI

URI parameter	Description
homeFolder	Link to My Documents

requests index page

Provides the outermost structure for the active request functionality. The requests index page displays the side menu and banner elements, and the tabbed property sheets defined by tabs. selectjobs.jsp uses the HTML code in <context root>\iportal\activePortal\private\jobs\selectjobscontent.jsp to display request data.

- Name**
- <context root>\selectjobs.do

<context root>\iportal\activePortal\private\jobs\selectjobs.jsp

Parameters Table 4-21 lists and describes the parameters for the requests index page. The requests index page also uses the common URI parameters.

Table 4-21 Parameters for request index URI

URI parameter	Description
applyFilter	Specifies whether to apply cbFail, cbSuccess, and filter to the current user session. applyFilter applies only to list pages, such as the completed jobs page.
cbFail	Specifies whether to list the failed jobs in the completed jobs page.
cbSuccess	Specifies whether to list the successful jobs in the completed jobs page.
channelName	Specifies the channel to which a job completion notice was sent. channelName applies only to the details page.
clearFilter	Clears the job name filter. clearFilter causes Actuate Information Console to retrieve job names from session cookies and to ignore cbFail and cbSuccess. clearFilter applies only to list pages, such as the completed jobs page.
filter	Specifies the job name filter. filter applies only to list pages, such as the completed jobs page.
jobID	Specifies the unique job identifier. jobID applies only to the details page.
resetFilter	Resets all filters to their default values. The default filter values are no filtering for job name, and listing all completed jobs, whether failed or successful. resetFilter applies only to list pages such as the completed jobs page.
subpage	Determines the content page. Possible values for subpage are: <ul style="list-style-type: none">■ _completed■ _detail■ _pending■ _running■ _scheduled _completed is the default content page.
userName	Specifies the name of the user who received the completed job notice. userName applies only to the detail page.

Used by Not applicable.

license page

Displays the License page, containing information about the Actuate Information Console version and options. Called when the user chooses the License tab on the Options page.

The default license page for Information Console is similar to Figure 4-6.



Figure 4-6 Information Console license page

Name	<context root>\portal\activePortal\private\options\license.jsp
Parameters	The about page uses the common URI parameters.
Used by	portal\activePortal\private\options\optionspage.jsp

list page

Lists files in a container, such as a channel or folder. There are three types of lists:

- channels
- channel contents
- filesfolders

channels list page

Lists the channels that the user subscribes to. Users can also subscribe or unsubscribe to channels from this page.

A channels list page looks like Figure 4-7. Users choose a channel name to see the contents of the channel.

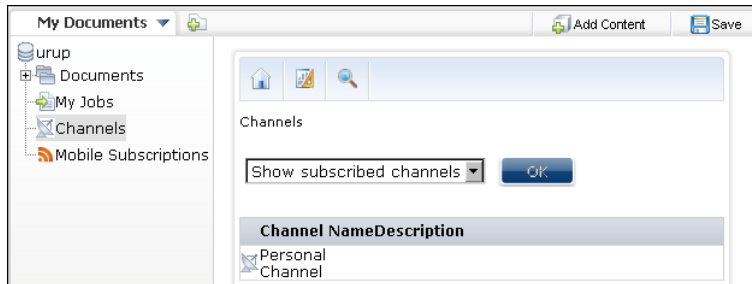


Figure 4-7 Channels list page

Name <context root>\selectchannels.do

Used by Not applicable.

channel contents list page

Lists the contents of a specified channel. You cannot access this page directly, but you can edit it to change its appearance. channelnoticelist.jsp uses the HTML code in <context root>\portal\activePortal\private\channels\channelnoticelistcontent.jsp to display the contents.

A channel contents list page looks like Figure 4-8.

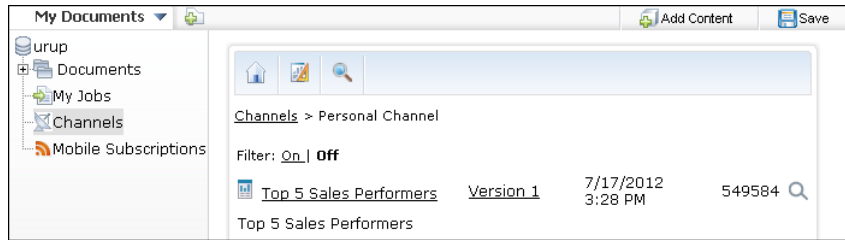


Figure 4-8 Channel contents list page

Users choose the file or version name to view the report document. Users choose the magnifying glass to view report details.

Name <context root>\selectjobnotices.do

Parameters Table 4-22 describes the parameter for the channel contents list page. The channel contents list page also uses the common URI parameters.

Table 4-22 Parameter for channel contents list URI

URI parameter	Description
__channel	The name of the channel to list

Used by Not applicable.

file and folder list page

Presents a list of objects that reside in the current working repository folder. Users request folder listings by choosing links on the reporting web page. The file and folder list page includes a filter section where users specify criteria for viewing report documents. For example, users select check boxes to indicate whether they want to view only the last version of a report document or to see report executable files and report documents.

When users access a repository for the first time, Actuate Information Console displays their home folder, if they have one, or the top folder in the repository. All files and folders in that folder that they have permission to view appear in the Actuate Information Console listing page. Users can specify a filter to choose the types of files to view.

The following are the sources that the file and folder list page uses to obtain the values for filters and the state of check boxes:

- URI parameters. See the following parameters section.
- Session attributes. Actuate Information Console uses session cookies to store the values that a user specifies. If the user browses the Actuate Information Console application, then returns to the listing page, the list page obtains the user's values from the session cookie if cookies are enabled. If the user chooses

another folder, that folder becomes the working folder, and the list page applies the same values that applied to the previous folder.

Table 4-23 lists and describes the session attribute variables.

Table 4-23 Session attribute variables

Session attribute	Description
AcFilesFoldersFilter	Contains the string specifying the files and folders viewing filter
AcFilesFoldersType Filter	Contains True if the user specified a filter, False otherwise
AcLastViapplicationd Folder	Contains the string specifying the last viapplicationd folder name

Name <context root>\getfolderitems.do

<context root>\portal\activePortal\private\filesfolders\filefolderlist.jsp

Parameters Table 4-24 lists and describes the parameters for the file and folder list page. The file and folder list page also uses the common URI parameters.

Table 4-24 Parameters for file and folder list URI

URI parameter	Description
applyFilter	If True, apply filter. If False, filter not applied.
filter	The filter specifying the file and folder names to list. Filter is a string. The default is "".
folder	The folder for which to list the contents. Folder name is a string. If no folder is specified, List uses the last working folder known for the session if cookies are enabled. If cookies are not enabled, List uses the user's home folder as specified in the user settings.
onlyLatest	If True, show only the latest version of a file if multiple versions exist. If False, show all versions of a file if multiple versions exist. The default is False.
resetFilter	Any non-null value for resetFilter causes the filter to return to its original state. Users can reset the filter by choosing the Default button on the listing page.
showDocument	If True, show all viewable documents. If False, do not show viewable documents. The default is True.
showExecutables	If True, show all report executables. If False, do not show report executables. The default is True.

Table 4-24 Parameters for file and folder list URI

URI parameter	Description
showFolders	If True, show all folders. If False, do not show folders. The default is True.

Used by Not applicable.

login banner page

Displays the Actuate Information Console web application banner. Banner elements include the company logo, system name, and help link.

Name <context root>\portal\activePortal\private\login_banner.jsp

Used by portal\activePortal\private\login.jsp

login page

Displays the Actuate Information Console login page for logging in to the Actuate Information Console web application. The login page includes the login banner page to display the Actuate Information Console application banner.

Name <context root>\login.do

<context root>\portal\activePortal\private\login.jsp

Parameters Table 4-25 lists and describes the parameters for the login page. The login page also uses the common URI parameters.

Table 4-25 Parameters for login page URI

URI parameter	Description
loginPostBack	False to display the login page and True to display the destination page instead of the login page if the login is successful.
targetPage	Specify a relative URI to which login redirects the user on successful login. The default is the file and folder list page.

Used by Not applicable.

logout page

Ends the user's Actuate Information Console session. The logout page gathers the user's session information, clears it, and returns the user to the login page.

Name <context root>\logout.do

Parameters Table 4-26 lists and describes the parameters for the logout page. The logout page also uses the common URI parameters.

Table 4-26 Parameters for logout page URI

URI parameter	Description
daemonURL	Contains the URI that accesses the Process Management Daemon, such as http://Server:8100.
user	The name of the user to log out. Either user or the common URI parameter authID must be specified. If authID is specified, user is ignored.

Used by Not applicable.

My dashboard page

A property sheet that supports specifying the default dashboard and resetting the layout of the default dashboard.

A My dashboard page looks like Figure 4-9.

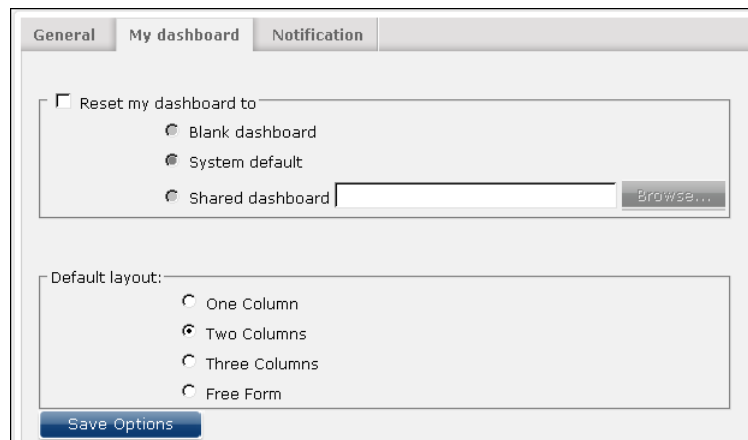
The image shows a web-based configuration window titled "My dashboard". It has three tabs: "General", "My dashboard" (which is active), and "Notification". Under the "My dashboard" tab, there are two main sections. The first section is "Reset my dashboard to" and contains three radio button options: "Blank dashboard", "System default", and "Shared dashboard". The "Shared dashboard" option is selected, and next to it is a text input field followed by a "Browse..." button. The second section is "Default layout:" and contains four radio button options: "One Column", "Two Columns" (which is selected), "Three Columns", and "Free Form". At the bottom of the window is a blue button labeled "Save Options".

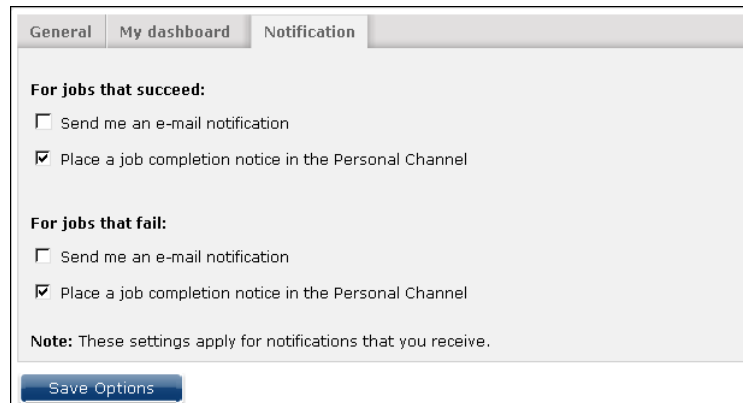
Figure 4-9 My dashboard page

Name <context root>\options.do
Used by iportal\activePortal\private\options\optionspage.jsp

notification page

A property sheet that supports specifying notification options for the current user. Notification options include whether to generate e-mail on completion of requests.

A notification page looks like Figure 4-10.



General My dashboard Notification

For jobs that succeed:

☐ Send me an e-mail notification

☒ Place a job completion notice in the Personal Channel

For jobs that fail:

☐ Send me an e-mail notification

☒ Place a job completion notice in the Personal Channel

Note: These settings apply for notifications that you receive.

Save Options

Figure 4-10 Notification page

Name <context root>\options.do
Used by iportal\activePortal\private\options\optionspage.jsp

options page

Updates the user options and settings on the server.

An options page looks like Figure 4-11.

Figure 4-11 Options page

Name <context root>\options.do

<context root>\portal\activePortal\private\options\options.jsp

Parameters Table 4-27 lists and describes the parameters for the options page. The options page also uses the common URI parameters.

Table 4-27 Parameters for options URI

URI parameter	Description
channelIcons	Specifies whether or not to display channel icons.
channels	Contains the string list of channels to which the user subscribes.
confirmKey	Contains the user's password.
docChanFilters	Specifies filters for viewing documents or channels.
email	Contains the user's e-mail address.
failComp	Indicates whether to generate completed request notifications for failed jobs. Enable to generate notifications for failed requests, Disable otherwise.
failEmail	Indicates whether to generate e-mail for failed requests. Set the value to "on" to enable or "off" to disable.

Table 4-27 Parameters for options URI

URI parameter	Description
newKey	Contains the user's new password.
newLocale	Contains the user's new locale.
newTimeZone	Contains the user's new time zone.
oldKey	Contains the user's old password.
redirect	Specifies the page to go to when user options update is complete.
requestFilters	Indicates whether to use filters for the Request page. Enable to use filters, Disable otherwise.
succComp	Indicates whether to generate completed request notifications for successful requests. Enable to generate notifications for failed requests, Disable otherwise.
succEmail	Indicates whether to generate e-mail for successful requests. Set the value to "on" to enable or "off" to disable.
userName	Contains the current user's name.
viewNewBrowser	Indicates whether to view documents in the current browser window or in a new browser window. Set the value to "on" to view documents in a new browser window or "off" to disable.

Used by Not applicable.

output page

Specifies report executable output data, such as the report headline and output file name. The output page appears only for scheduled report job and Run and Save report job submissions. Users access Output by choosing Save As.

An output page looks like Figure 4-12.

Figure 4-12 Output page

Name <context root>\portal\activePortal\private\newrequest\output.jsp

Parameters Table 4-28 lists and describes the parameters for the output page. The output page also uses the common URI parameters.

Table 4-28 Parameters for output URI

URI parameter	Description
headline	Specifies the headline for the report.
ifExists	Specifies the file replacement policy. Values are Create and Replace. If ifExists is Create, Actuate Information Console creates a new version. If ifExists is Replace, Actuate Information Console replaces the existing version.

Table 4-28 Parameters for output URI

URI parameter	Description
outputFolderType	Specifies the report output's folder type. Values are personal and absolute. If outputFolderType is personal, the output is placed in the user's personal folder. If outputFolderType is absolute, the user specifies the full path name for the output by either typing the path or using the Browse button.
outputName	Specifies the name of the output file.
versionName	Specifies the version name.

Used by iportal\activePortal\private\newrequest\newrequestpage.jsp

page not found page

Displays an error message when Actuate Information Console cannot find the page that a user specifies. This page is an Information Console page only.

Name <context root>\iportal\activePortal\errors\pagenotfound.jsp

Used by Not applicable.

parameters page

Displays report job parameters. Parameters include the headline, output file name, and report executable (.rox) file name. Users access the parameters list by choosing Parameters.

The parameters page looks like Figure 4-13.

Figure 4-13 Parameters page

Name <context root>\iportal\activePortal\private\newrequest\parameters.jsp

Used by iportal\activePortal\private\newrequest\newrequestpage.jsp

pending page

Lists all jobs that are currently awaiting execution.

Name <context root>\portal\activePortal\private\jobs\pendingjob.jsp

Parameters The pending page uses the common URI parameters.

Used by portal\activePortal\private\jobs\selectjobscontent.jsp

ping page

The ping page tests whether a specific component of the reporting environment is operational, and optionally retrieves other diagnostic information about the component. You can test the following components of the reporting environment:

- Information Console itself
- The Caching service
- The Encyclopedia service
- The Factory service
- The Integration service
- The Message Distribution service (MDS)
- The View service
- An Actuate open server driver

If a component is not operational, Actuate BIRT iServer returns an error message. If a component is operational, the response depends on the ping page parameters. For example, you can request a simple time stamp that shows the time elapsed between the time that a component receives the request and the time that it returns a reply, as shown with the following URI:

`http://seamore:8700/portal/ping.do?destination=EE&mode=trace`

generates the following response:

```
18:03:23.100: MDS(seamore) received Ping message
18:03:23.100: MDS(seamore) forwarding Ping request to node seamore
18:03:23.100: EncycEngine(seamore) received Ping message
EncycEngine(seamore): Echoing 0 bytes of payload data
18:03:23.100: EncycEngine(seamore) replying to Ping message.
    Elapsed=0 ms
18:03:23.100: MDS(seamore) received Ping reply from node seamore.
    Roundtrip= 0 ms
18:03:23.100: MDS(seamore) replying to Ping message. Elapsed=0 ms
```

You also can request more detailed information. A ping request to the MDS has no security restrictions. For all other components, the request is subject to Encyclopedia volume authentication. The user must be an Encyclopedia volume administrator or a user with the Operator security role.

Name <context root>\ping.do

Parameters Table 4-29 lists and describes the parameters for the ping page. The ping page also uses the common URI parameters.

Table 4-29 Parameters for ping URI

URI parameter	Description
action	<p>Specifies the action to take at the destination. Valid values are:</p> <ul style="list-style-type: none">■ Echo—Echoes data specified by the payloadSize parameter. Echo is the default action.■ ReadFile—Opens a specified Encyclopedia volume file, reads its content, and closes the file. Destination must be EE, FS, or VS.■ WriteFile—Creates a temporary file in a partition, writes a specified number of bytes, closes the file, and deletes it. Destination must be EE or FS.■ Connect—Connects to a data source. <p>If you do not specify a value, the destination component responds to the request without taking any other actions.</p>
destination	<p>The reporting environment component to test. Valid values are:</p> <ul style="list-style-type: none">■ AP (Information Console)■ MDS (Message Distribution service)■ EE (Encyclopedia Engine)■ FS (Factory service)■ VS (View service)■ AIS (Actuate Integration service)■ ACS (Actuate Caching service) <p>AIS and ACS only support the Echo action.</p> <p>Except when AP is specified as destination, Actuate Information Console sends a ping request to the Actuate BIRT iServer and passes on the destination as the ping request's destination parameter.</p> <p>The default value is AP.</p>

(continues)

Table 4-29 Parameters for ping URI (continued)

URI parameter	Description
filename	When action=ReadFile, this parameter is required to indicate the Encyclopedia volume file to read. If you ping an open server driver, filename specifies the executable file to prepare for execution.
mode	<p>Specifies the level of detail in the ping response. Valid values are:</p> <ul style="list-style-type: none"> ■ Concise—Returns the elapsed time between a component's receipt of the request and the time the component sends a reply. ■ Normal—Returns the names of components in the test path and the time stamps of the request entering and leaving each component. This is the default mode. ■ Trace—Returns a time stamp at times when the request enters and leaves major subcomponents of the component being tested. For example, a request to a node running the Encyclopedia service can provide a time stamp for times when the request enters and leaves the process queue. <p>A ping request at the trace level also can return diagnostic information other than timing. For example, a request to test writing a temporary file to a partition can return the amount of free disk space on the partition.</p>
partitionName	Specifies the name of the Encyclopedia partition on which to create the temporary file. Used only if the value of action is WriteFile.
payloadSize	Length of payload string in number of characters that Actuate Information Console should generate. Used only if the value of action is Echo.
processID	Specifies the process ID of the Factory or View service to test. Used with the server parameter.
server	<p>Specifies which instance of a Factory service or View service to test. Works with the processID parameter. To test all available instance of the Factory or View service, use an asterisk (*).</p> <p>If you do not specify server, the Actuate BIRT iServer load balancing mechanism allocates an available instance of the requested service to respond to the ping request.</p>

Used by Not applicable.

print page

Prints all or part of a report document in PDF format. You need an Acrobat Reader to view the report document on screen.

The following URI creates a PDF of the first two pages of the forecast report document and opens a print dialog:

```
\iportal\viewer\print.jsp?name=\forecast.roi&range=1-2
```

Name <context root>\iportal\activePortal\viewer\print.jsp

Parameters Table 4-30 lists and describes the parameters for the print page. The print page also uses the common URI parameters. If the page and range parameters are both specified, range is used. If neither page nor range is specified, Information Console creates a PDF of the entire report document and opens a print dialog.

Table 4-30 Parameters for print URI

URI parameter	Description
connectionHandle	Optional identifier returned by the Actuate BIRT iServer.
ID	The ID of the report document to print.
name	The name of the report document to print. This parameter is ignored if ID is also specified.
page	The page to print.
range	A range of pages to print. Separate pages and page ranges with commas, such as 1-3,15,21-25.

Used by Not applicable.

privileges page

Assigns privileges to a file or folder. Filefoldersprivilege.do uses the HTML code in <context root>\iportal\activePortal\private\filefolders\filefolderlist.jsp to set the privileges. The following URI displays the privilege page for the hotgraph report executable in the Training folder:

```
\iportal\filefoldersprivilege.do?name=\Training\hotgraph.sox
```

Name <context root>\filefoldersprivilege.do

Parameters Table 4-31 lists and describes the parameter for the privileges page. The privileges page also uses the common URI parameters.

Table 4-31 Parameters for privileges URI

URI parameter	Description
name	File or folder name to set privileges for

Used by `iportal\activePortal\private\common\popupmenu.jsp`
`iportal\activePortal\private\filesfolders\filedetailcontent.jsp`

running page

Lists all jobs that were executing when the running page was last refreshed. The list is not live. To view the current list, the user must refresh the browser. Users access the running jobs list by choosing Running.

The running page looks like Figure 4-14.

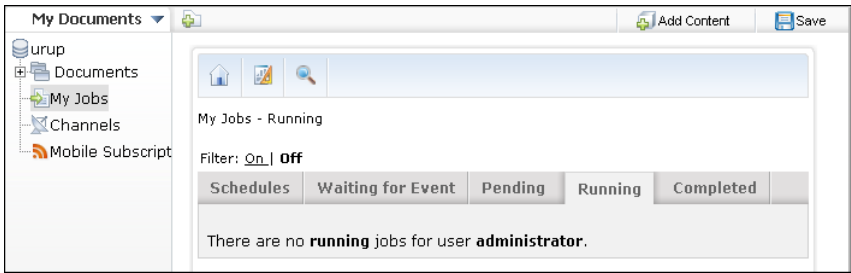


Figure 4-14 Running page

Parameters The running page uses the common URI parameters.

Name `<context root>iportal\activePortal\private\jobs\runningjob.jsp`

Used by `iportal\activePortal\private\jobs\selectjobscontent.jsp`

save as page

Supports downloading the current report executable (.rox) job output, saving it in PDF, PowerPoint, or RTF format, or exporting report data to Excel. Users choose Download in the DHTML Viewer toolbar to access the save as page and save report output in PDF, PowerPoint, or RTF format or to export report data to Excel.

The save as page looks like Figure 4-15.

Export Report To:

☒ PDF

PDF Quality: 100
Split Large Pages: Default
Page Width:
Page Height:

☐ Excel Data
☐ Excel Display
☐ PowerPoint
☐ RTF
☐ Fully Editable RTF

Tips:

1. Excel Data format is good for data manipulation. It was designed for tabular and listing reports.
2. Use PowerPoint format for fully editable presentations of reports with charts and dynamic text.
3. Fully Editable RTF format is good for multi-control editing, but creates significantly larger files than RTF format.
4. PDF Quality level 100 gives the lowest image quality but the smallest PDF file size, and 300 gives the highest image quality but the largest PDF file size.

Page Range:

☐ All
☒ Current page
☐ Pages:

Enter page numbers and continuous page ranges separated by commas. For example: 1,3,5-12.

View Report

Save Report

Figure 4-15 Save as page

Name <context root>\portal\activePortal\viewer\saveas.jsp

Parameters Table 4-32 lists and describes the parameters for the save as page. The save as page also uses the common URI parameters.

Table 4-32 Parameters for save as URI

URI parameter	Description
componentID	Locates the page containing the specified component ID.
embSrvRequester	The prefix for an embedded object to retrieve the object from the report output. Embedded objects include hyperlinks, cascading style sheets, static objects such as images, and dynamic objects such as charts.
encoding	The character encoding, such as UTF8.
mode	The page navigation mode. Values are First, Last, Previous, or Next.
objectID	The ID of the report object being searched.
page	The page to view.
range	A range of pages to retrieve. Separate pages and page ranges with commas, such as 1-3, 15, 21-25.
reportletMaxHeight	The maximum height in points of Reportlets. Used by the Reportlets feature.

(continues)

Table 4-32 Parameters for save as URI (continued)

URI parameter	Description
scalingFactor	The size of the report document in the browser, such as 100 (full size) or 50 (half size).
searchList	The list of name-value pairs that specify the page or pages that contain the components to search. The format is: <code>&<component name>=<value></code> <code>[&<component name> = <value>] ...</code> where <component name> is the fully qualified name of the component on which the search condition is based. Do not enter the “searchList” parameter name; enter only the component/value pairs. For example, <code>titleframe::txtname:include=*</code> .
type	If name is specified, the type of object to search, such as ROI.
userAgent	The user’s browser, such as Mozilla/4.0.
version	The report document’s version number. If Version is not specified, the latest version of the report document is used.

Used by iportal\activePortal\private\newrequest\newrequestpage.jsp

schedule page

Supports specifying report executable file run schedules. The schedule page applies only to scheduled report job requests.

Schedule properties include data and time for running the request, recurring schedules to run a report job on a regular basis, or whether to run the report job immediately.

The Information Console schedule page is similar to Figure 4-16.

`<context root>\js\calendar.js` provides calendar functionality for Information Console. Note that date and time field lengths are hard-coded in the schedule page.

Figure 4-16 Information Console schedule

Name <context root>\iportal\activePortal\private\newrequest\schedule.jsp

Parameters Table 4-33 lists and describes the parameters for the schedule page. The schedule page also uses the common URI parameters.

Table 4-33 Parameters for schedule URI

URI parameter	Description
jobName	The name of the request being submitted.
onceDate	If scheduleType is once, specify the date on which to run the report job.
onceTime	If scheduleType is once, specify the time at which to run the report job.
recurringDay	The day on which to run the request on a regular basis. Values are the day of the week, EVERYDAY, FIRST_DAY_OF_THE_MONTH, LAST_DAY_OF_THE_MONTH.
recurringTime	If scheduleType is recurring, specify the time at which to run the report job.
scheduleType	Specify the schedule type. Values are immediate, once, and recurring.

Used by iportal\activePortal\private\newrequest\newrequestpage.jsp
iportal\activePortal\private\query\runpage.jsp

scheduled job page

Lists all jobs that activate at a specified date and time but are not yet active.

Name <context root>\iportal\activePortal\private\jobs\scheduledjob.jsp

Parameters The scheduled job page uses the common URI parameters.

Used by Not applicable.

search folders page

Recursively searches from the current folder for files and folders whose names match the search string.

Name <context root>\searchfiles.do

Parameters Table 4-34 lists and describes the parameters for the search folders page. The search folders page also uses the common URI parameters.

Table 4-34 Parameters for search folders URI

URI parameter	Description
folder	Folder name to start the search from. The default is the current location, as shown in the breadcrumb.
searchFilter	The name to search for. Expressions and wildcards are allowed. For more information about search expressions, see <i>Working with Actuate e.Reports</i> .

For example, the following Information Console URL searches in the current folder and all subfolders for files or folders whose names begin with the string Cust:

`http://localhost:8700/iportal/searchfiles.do?searchFilter=Cust*`

A search results page looks like Figure 4-17.

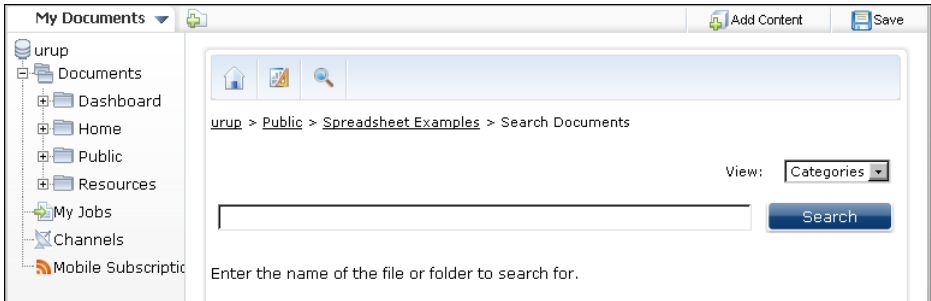


Figure 4-17 Search results

Used by Not applicable.

submit job page

Submits a scheduled report job for a report executable or Actuate Query to the server. There is no user interface to the submit job page. submitjobstatus.jsp uses the HTML code in <context root>\iportal\activePortal\private\newrequest\submitjobstatuspage.jsp to display the new request information.

For reports that accept run-time parameters, set the parameter in the URL by adding an ampersand (&), the parameter name, and an equal (=) sign, followed by the parameter value in quotation marks.

Name <context root>\submitjob.do

<context root>\iportal\activePortal\private\newrequest\submitjobstatus.jsp

Parameters Table 4-35 lists and describes the parameters for the submit job page. The submit job page also uses the common URI parameters. All other parameters are passed to the report executable as report parameters. Report parameters are case-sensitive. Specify them exactly as defined in the report design.

The submit job page also accepts dynamic filter parameters for BIRT Reports in the URL, but the value of the parameter must form a complete expression, such as &Territory=([Territory] = "EMEA").

Table 4-35 Parameters for submit job URI

URI parameter	Description
__accessToGrant	Grants read or secure read privileges to those roles that have permission to view the report document. For users to view only the parts of the document matching an access control list (ACL), grant Secure Read access. Otherwise, grant Read access to enable users to view the whole document. This parameter requires the __channels, __exclude, and invokeSubmit=true parameters, even if you use no value for them. Use the __exclude parameter with this parameter to exclude specific users from getting the privilege. Use the __channels parameter to grant read privileges to channels and notify them.
__ageDays	Used with __ageHours to determine how long output objects exist before they are deleted. Use only if __archivePolicy is set to age. __ageDays can be any positive number.
__ageHours	Use with __ageDays to determine how long output objects exist before they are deleted. Use only if __archivePolicy is set to age. __ageHours can be any positive number.

(continues)

Table 4-35 Parameters for submit job URI (continued)

URI parameter	Description
<code>__archiveBeforeDelete</code>	Indicates whether to archive the output objects of the request before deleting them, according to <code>__archivePolicy</code> 's setting. Set this parameter to <code>True</code> to archive objects before deleting them. The default value is <code>False</code> . This parameter has no effect if <code>__archivePolicy</code> is set to <code>folder</code> .
<code>__archivePolicy</code>	The archive policy to implement for the objects created as output for the request. Values are <code>folder</code> , <code>age</code> , and <code>date</code> . Set this parameter to <code>folder</code> to use the archive policy already set for the folders to which the output is distributed, to <code>age</code> to delete objects older than a specific time period, or to <code>date</code> to delete objects on a specific date.
<code>__dateToDelete</code>	The date on which to delete the output objects of the current request. Use only if <code>__archivePolicy</code> is set to <code>date</code> . <code>__dateToDelete</code> must be a date in a locale-specific format. The default format is <code>mm/dd/yyyy</code> .
<code>__executableName</code>	The name of the executable file for this request.
<code>folderType</code>	Specifies the destination folder type for the report. <code>Absolute</code> indicates the repository root folder, <code>/</code> . <code>Personal</code> indicates the current user's home folder. Default is <code>Personal</code> .
<code>__format</code>	Output file format. The ROI format of an e.report is always created first, and then converted to the specified format if that format is not ROI. Optionally, specify keeping the ROI output with the <code>__keepROIIfSucceeded</code> and <code>__keepROIIfFailed</code> options. Format values are: <ul style="list-style-type: none"> ■ AFP—an Advanced Function Printing format used primarily as a print stream. ■ ExcelData—an Excel spreadsheet format to use for basic tabular or listing report documents. The appearance can differ from the original report document. Do not use this format for a complicated report document. Potential issues in this format include images and graphs that do not appear, missing background colors for frames and flows, and imprecise component positioning. ■ ExcelDisplay—an Excel spreadsheet format that appears as much like an Actuate report document as possible. ■ PDF—an Adobe Acrobat-readable Portable Document Format file. ■ ROI—a report object instance (<code>.roi</code>) file, which a user can view using the DHTML Viewer. This is the default format. ■ RTF—a rich text format (<code>.rtf</code>) file. The report document's visual layout is similar to the DHTML Viewer layout.

Table 4-35 Parameters for submit job URI (continued)

URI parameter	Description
<code>__format</code> (continued)	<ul style="list-style-type: none"> ■ RTFFullyEditable—a rich text format (.rtf) file with more flexibility for manipulating output, such as moving or multiple lines at the same time. Produces a larger RTF file than the RTF format. ■ Actuate Information Console truncates report documents in the ExcelData and ExcelDisplay formats if they exceed the Maximum Number of Pages Convertible To Excel parameter value.
<code>__headline</code>	A descriptive tag line for a report document. Appears on the Channel Contents page. Use the character string %20 to represent spaces in the headline string.
<code>__ifExists</code>	Indicates whether to overwrite an existing or create a new file, up to an optional limit. Values are: <ul style="list-style-type: none"> ■ create—creates a new output file. ■ create[n]—creates a new output file up to n versions. For example, to create no more than seven versions, use create7. ■ replace—overwrite any existing output.
<code>invokeSubmit</code>	Controls whether the browser is redirected to the parameter screen or whether the report job is scheduled immediately. If True, the report job is scheduled without displaying the parameters. If False, the parameters are displayed. False is the default.
<code>__jobName</code>	The name for the job to submit.
<code>__keepROIIf Succeeded</code>	Used with the <code>__format</code> parameter. Specifies whether to keep the ROI after successfully generating a non-ROI format. The default value is False, which deletes the ROI.
<code>__keepROIIfFailed</code>	Used with the <code>__format</code> parameter. Specifies whether to keep the ROI if a non-ROI format is selected for report generation and the conversion step fails. The default value is False, which deletes ROI.
<code>notification Attachment</code>	Sets the format of the attachment sent with e-mail notification for this job. Accepts the same values as <code>__Format</code> .
<code>notificationSupported</code>	Specifies whether to notify users who have notification disabled. True sends notification and disregards user preferences. Default value is False.
<code>notify</code>	Activates e-mail notification for the job.

(continues)

Table 4-35 Parameters for submit job URI (continued)

URI parameter	Description
<code>--onceDate</code>	Required for once schedules. Specify the date on which to run the report job, for report jobs with <code>--scheduleType</code> of once. Must be in the appropriate format for your locale, such as mm/dd/yyyy for the U.S. locale. The current date is the default.
<code>--onceTime</code>	Required for once schedules. Specify the time at which to run the report job, for report jobs with <code>--scheduleType</code> of once. Must be in the appropriate format for your locale, such as "hh:mm a" for the U.S. locale. The current time is the default.
<code>--outputName</code>	Specifies a name for the report output document.
<code>outputName</code>	Specifies a name for the report output document for the e-mail notification.
<code>outputFormat</code>	Optional parameter that appends a file extension to the <code>outputName</code> . Do not use a period in the value of this parameter, a period is inserted automatically before the file extension.
<code>postback</code>	Forces the browser not to display parameters. Set to False to display parameters. Do not set <code>postback</code> to True with <code>invokeSubmit</code> also set to True.
<code>--priority</code>	Specifies the job submission priority. Values are a number from 1 to 1000, High (800), Medium (500), and Low (200). Do not use with <code>--priorityValue</code> .
<code>--priorityValue</code>	Specifies a number corresponding to the job submission priority. Do not use with <code>--priority</code> .
<code>--progressive</code>	Indicate whether to display the report document after it generates. If False, the report document displays after it generates. If True, the report document displays progressively, as it generates. Applies only to run report jobs.
<code>--recurringDay</code>	Specifies the scheduled recurring day on which to run the report job. Applies only to scheduled report jobs.
<code>--recurringTime</code>	Required for recurring schedules. Specify the time at which to run the report job. Set only if report jobs <code>--scheduleType</code> is recurring. Must be in the appropriate format for your locale, such as hh:mm:ss for the U.S. (enu) locale.
<code>--redirect</code>	Specifies a relative or absolute URI to go to after <code>do_executereport</code> submits the report job. The default is <code>Submittedjob_Status</code> .

Table 4-35 Parameters for submit job URI (continued)

URI parameter	Description
__schedulePeriod	Required for recurring schedules. Specify how often to run the report job, and on which days. Choose a day of the week. __schedulePeriod values are Every Day, Weekdays, Mondays, Tuesdays, Wednesdays, Thursdays, Fridays, Saturdays, Sundays, First Day of the Month, Last Day of the Month. All values are case-sensitive. Every Day or Weekdays. Set only if __scheduleType is recurring.
__scheduleType	Specify the type of schedule: immediate, once, or recurring. Immediate is the default.
__serverURL	Contains the URI that accesses the JSP engine, such as http://Services:8700.
__timeToDelete	Specifies a time at which to delete an archived report document. Applies only to scheduled report jobs.
__versionName	Contains a string value for the new version name of the job's report document output. The value can include a date/time expression enclosed in braces, {}, to ensure a unique version name.
__volume	Contains a string value specifying the volume for the job's report document output.

For example, the following URL schedules the Sales By Territory.rptdesign report to run once on the September 16, 2010 with the Territory run-time parameter set to Japan:

```
http://localhost:8700/iportal/  
submitjob.do?__requesttype=scheduled&__executableName=%2fPublic  
%2fBIRT%20and%20BIRT%20Report%20Studio%20Examples%2fSales%20by%  
20Territory%20erptdesign%3b1&userid=administrator&__scheduleType  
=once&__onceDate=09/16/2010&__onceTime=1:55  
pm&Territory="Japan"&invokeSubmit=True
```

Used by iportal\activePortal\private\filesfolders\filefolderlistcontent.jsp
 iportal\activePortal\private\newrequest\newrequestpage.jsp

submit page

Copies or reruns a completed query using the Actuate Query Wizard. execute.jsp uses the HTML code in <context root>\iportal\activePortal\private\query\runpage.jsp to display the query data.

Name <context root>\query\submit.do

<context root>\portal\activePortal\private\query\execute.jsp

Parameters Table 4-36 describes the parameter for the submit page. The submit page also uses the common URI parameters.

Table 4-36 Parameter for submit URI

URI parameter	Description
__executableName	The full path name of the DOV or DOX file to use for the query

Used by Not applicable.

view cube page

Displays the Actuate Analytics Cube Viewer to view or analyze a data cube in an Encyclopedia volume. When a cube view is accessed, the latest version of the data cube is generated from the cube design that the cube view depends upon. This lets you create views on the cube and use the views as bookmarks to analyze the cube.

Many settings for the Cube Viewer are taken from the Actuate Information Console configuration parameters. These settings include the viewer's height and width, the default experience level, and whether or not users can save new cubes. The names of all of the configuration parameters that apply to the Cube Viewer begin with ANALYTICS.

The Cube Viewer downloads the entire cube file from the Encyclopedia volume to allow analysis on the cube. The upper limit on the cube size that can be downloaded depends on the resources available on your client machine, and can be less than necessary to download a large cube. Actuate recommends partitioning the data to be analyzed based on the end user role, and using a summary and several detail cubes to support analysis of large cubes.

Name <context root>\viewcube.do

Parameters Table 4-37 describes the parameter for the view cube page. The view cube page also uses the common URI parameters.

Table 4-37 Parameter for view cube URI

URI parameter	Description
name	Name of the cube or the view in the Encyclopedia. The file name specified must include the folder path and file version.

For example, the following Information Console URI executes the Actuate Analytics Cube Viewer to view a cube using the cube name:

`http://druid:8700/portal/viewcube.do?name=/sales/forecast.cb4`

Used by Not applicable.

Actuate BIRT Viewer URIs reference

To view and interact with Actuate BIRT reports, you use the Actuate BIRT Viewer servlet. All BIRT Viewer options and varieties use the same URI. For detailed information about the BIRT Viewer servlet URI, see *Working with Actuate BIRT Viewers*.

Actuate Viewer URIs reference

This section provides the detailed reference for Actuate DHTML Viewer URIs, used to view and interact with Actuate e.reports. In the definitions, <context root> represents the name of your Actuate Information Console context root, initially portal.

Table 4-38 lists the topics this section covers and the file names discussed in each topic. All pages are under the Information Console context root.

Table 4-38 Actuate BIRT Viewer pages

Topic	Information Console file
request search page	portal\activePortal\viewer\requestsearch.jsp
search frame page	portal\activePortal\viewer\searchframe.jsp
search report page	portal\activePortal\viewer\searchreport.jsp
search toolbar page	portal\activePortal\viewer\searchtoolbar.jsp
view default page	portal\activePortal\viewer\viewdefault.jsp
view frame set page	portal\activePortal\viewer\viewframeset.jsp
view navigation page	portal\activePortal\viewer\viewnavigation.jsp
view TOC page	portal\activePortal\viewer\viewtoc.jsp

request search page

Displays the search request form and invokes the search frame page to perform the search.

The search request form looks like Figure 4-18.



Figure 4-18 Search request form

Name <context root>\portal\activePortal\viewer\requestsearch.jsp

Used by viewer\searchframe.jsp
viewer\viewframeset.jsp

search frame page

Processes report document search conditions. The request search page, the search report page, and the view frame set page use the search frame page.

Name <context root>\portal\activePortal\viewer\searchframe.jsp

Used by viewer\requestsearch.jsp
viewer\searchreport.jsp

search report page

Searches a report document based on criteria that the user specifies in the search form, obtains search results, and presents the results to the user in the browser.

Name <context root>\portal\activePortal\viewer\searchreport.jsp

Parameters Table 4-39 lists and describes the parameters for the search report page. The search report page also uses the common URI parameters.

Table 4-39 Parameters for search report URI

URI parameter	Description
connectionHandle	An identifier for temporary report documents only.
enableColumnHeaders	Determines whether the search results include an initial record containing the names of the selected columns. Values are True to include the column headers in the search results, False to omit the headers. The default is True. This option applies only if format is CSV.
format	The format of the search data: XMLDisplay, CSV, and TAB. The report document encoding is used for the results file. If the report document encoding is UCS2, the CSV or TAB results file uses UTF8 encoding.
hits	The end value for the range for the search results.
ID	The ID of the report document to search.
name	The name of the report document to search. This parameter is ignored if objectID is also specified.
searchList	<p>The search criteria. The format is:</p> <pre><class>[.<variable>] [:include :exclude]=[value]</pre> <p>or:</p> <pre><SearchTag>[:include :exclude]=[value]</pre> <p>Do not type the searchList parameter name. Type only search criteria value pairs. The search criteria components are:</p> <ul style="list-style-type: none">■ <class> is the fully qualified name of a component class, such as the control component class. When creating a searchReport URI to embed in a report document or web page, examine the report document’s ROD file in Actuate e.Report Designer Professional to determine the fully qualified name of the component class to search for.■ <SearchTag> is the value of the SearchTag property, if specified in the component. To create a searchReport URI to embed in a report document or web page, examine the report document’s ROD file in Actuate e.Report Designer Professional to determine the SearchTag property of the component on which to search. If the component has a SearchTag, use the SearchTag not the component name. For more information about searching and SearchTags, see <i>Developing Reports using e.Report Designer Professional</i>.

(continues)

Table 4-39 Parameters for search report URI (continued)

URI parameter	Description
searchList	<i>(continued)</i> <ul style="list-style-type: none">■ <value> is the value or value expression for which to search. If not specified, the search uses the empty string, "". Value expressions can include relational operators, logical operators, and metacharacters, such as <, >, !, *. If the value expression contains special characters (>, <, &, ;, @, #) or has one or more spaces at the beginning or end of the expression, enclose the expression in quotation marks. For example, the value expression, C3=">10" specifies a search for a string containing the characters ">10" not a search for all C3 values greater than 10. ContentField="6*" specifies a search for a string containing the characters "6*" not a search for all ContentField values beginning with the character 6.■ :include :exclude are optional selection modifiers. These modifiers determine whether a specified select criterion, or component instance, is included in the search results. Specify :include to include the component instance in the search result. Specify :exclude to use the search criteria for the component class, but do not display the found instances. For example, if you build a URI with the select modifier EmpSal:exclude=>0, the search uses the value EmpSal > 0 as a select criterion, but does not return values for EmpSal in the search results. The modifier :include is the default.
startingPoint	The start value of the range for the search results.
type	The type of the object in which to search, such as ROI.
useQuoteDelimiter	Determines whether to enclose the search results in double quotes. True encloses each data item with double quotes in the search results, False does not. The default is True. Actuate ignores this option if format is not CSV.
version	The report document's version number. If you do not specify a version, Actuate uses the latest version of the report document.

Used by Not applicable.

search toolbar page

Builds and displays the search toolbar. The search toolbar appears at the top of the search form, as shown in Figure 4-19.



Figure 4-19 Search toolbar

Name <context root>\portal\activePortal\viewer\searchtoolbar.jsp

Used by viewer\searchframe.jsp
viewer\searchreport.jsp
viewer\viewframeset.jsp

view default page

Displays the e.report document with the navigation bar in the browser according to the user’s preferences and privileges, set in User Preferences. Either objectID or name must be specified. If an illegal set of attribute values are specified, or there is a processing error, an exception is thrown.

Name <context root>\portal\activePortal\viewer\viewdefault.jsp

Parameters Table 4-40 lists and describes the parameters for the view default page. The view default page also uses the common URI parameters.

Table 4-40 Parameters for view default URI

URI parameter	Description
connectionHandle	An identifier for temporary report documents only.
embSrvRequester	The prefix for an embedded object to retrieve the object from the report document. Embedded objects include hyperlinks, cascading style sheets, static objects such as images, and dynamic objects such as charts.
ID	The ID of the report document being searched.
mode	The page navigation mode. Values are First, Last, Previous, or Next.
name	The name of the report document to be viewed. This parameter is ignored if objectID is also specified.
page	The page number to view.
scalingFactor	The size of the report document in the browser, such as 100 (full size) or 50 (half size).

(continues)

Table 4-40 Parameters for view default URI (continued)

URI parameter	Description
searchCriteria	<p>The list of name-value pairs that uniquely identify a page or pages containing components on which to search. The format is:</p> <pre>&<component name>=<value> [&<component name>=<value>]...</pre> <p>where <component name> is the fully qualified name of the component on which the search condition is based.</p> <p>The & and = must be encoded to %26 and %3d. Do not enter the searchList parameter name; enter only the component/value pairs. For example, %26titleframe::txtname:include%3d*.</p> <p>searchCriteria finds the first matching component in the report document. Because the search is a component-based search and not page-based, it returns the start of the section or component that matches the search criteria rather than the page showing those matching values.</p>
showSearch	<p>If True, shows the search frame the first time the page is accessed. The default is False.</p>
showTOC	<p>If True, shows the TOC the first time the URL is accessed. The default is False.</p>
userAgent	<p>The user's browser, such as IE/5.5.</p>
version	<p>The report document's version number, if the name option is specified. If unspecified, the latest version of the report document is used.</p>

Used by Not applicable.

view frame set page

Displays the e.report document along with the navigation bar in the browser. The following example displays the report document Msbargph.roi:

```
http://infoconsole:8900/iportal/viewer/viewframeset.jsp
?name=/msbargph.roi;l&__vp=Seamore
```

Name <context root>\portal\activePortal\viewer\viewframeset.jsp

Parameters Table 4-41 lists and describes the parameters for the view frame set page. The view frame set page also uses the common URI parameters.

Table 4-41 Parameters for view frame set URI

URI parameter	Description
# - version	The report document's version number is indicated by a semicolon and an integer immediately following the file name. For example, msbargph.roi;1 indicates version 1 of msbargph.roi. If version is not specified, the latest version of the report document is used.
connectionHandle	An identifier for temporary report documents only.
closeX	Determines whether the parent window that generated the viewframeset request is closed when viewframeset.jsp is closed. The default value is False, which does not close the parent window when this window is closed.
embSrvRequester	The prefix for an embedded object to retrieve the object from the report document. Embedded objects include hyperlinks, cascading style sheets, dynamic objects such as charts, and static objects such as images.
ID	The ID of the report document being searched.
mode	The page navigation mode. Values are First, Last, Previous, or Next.
name	The name of the report document to be viewed. This parameter is ignored if ID is also specified.
page	The page to view.
scalingFactor	The size of the report document in the browser, such as 100 (full size) or 50 (half size).

(continues)

Table 4-41 Parameters for view frame set URI (continued)

URI parameter	Description
searchCriteria	<p>The list of name-value pairs that uniquely identify a page or pages containing components on which to search. The format is:</p> <pre>&<component name>=<value> [&<component name>=<value>] ...</pre> <p>where <component name> is the fully qualified name of the component on which the search condition is based. The & and = must be encoded to %26 and %3d. Do not enter the searchList parameter name; enter only the component/value pairs. For example, %26titleframe::txtname:include%3d*.</p> <p>searchCriteria finds the first matching component in the report document. Because the search is a component-based search and not page-based, it returns the start of the section or component that matches the search criteria rather than the page showing those matching values.</p>
showSearch	If True, shows the search frame the first time the page is accessed. The default is False.
showTOC	If True, shows the TOC the first time the URL is accessed. The default is False.
userAgent	The user's browser, such as Mozilla/4.0.

Used by iportal\activePortal\private\filesfolders\filefolderlistcontent.jsp
iportal\activePortal\viewer\viewdefault.jsp
iportal\activePortal\viewer\viewreport.jsp

view navigation page

Displays the viewer navigation toolbar for the Actuate e.Report DHTML Viewer.

Name <context root>iportal\activePortal\viewer\viewnavigation.jsp

Parameters Table 4-42 lists and describes the parameters for the view navigation page. The view navigation page also uses the common URI parameters.

Table 4-42 Parameters for view navigation URI

URI parameter	Description
name	The name of the report document to be viewed. This parameter is ignored if objectID is also specified.

Table 4-42 Parameters for view navigation URI

URI parameter	Description
objectID	The ID of the report document to display.
page	The page number to view.
scalingFactor	The size of the report document in the browser, such as 100 (full size) or 50 (half size).
searchCriteria	<p>The list of name-value pairs that uniquely identify a page or pages containing components on which to search. The format is:</p> <pre>&<component name>=<value> [&<component name>=<value>] ...</pre> <p>where <component name> is the fully qualified name of the component on which the search condition is based.</p> <p>The & and = must be encoded to %26 and %3d. Do not enter the searchList parameter name; enter only the component/value pairs. For example, %26titleframe::txtname:include%3d*.</p> <p>searchCriteria finds the first matching component in the report document. Because the search is component-based search and not page-based, it returns the start of the section or component that matches the search criteria rather than the page showing those matching values.</p>
type	If name is specified, the type of object to search, for example ROI.
userAgent	The user's browser, such as IE/8.0.
version	The report document's version number, if the name option is specified. If unspecified, the latest version of the report document is used.

Used by viewer\searchreport.jsp
viewer\viewframeset.jsp

view TOC page

Displays the report document's table of contents (TOC). Users view a report document's table of contents by choosing TOC on the toolbar as shown in Figure 4-20.



Figure 4-20 TOC toolbar

The content of the table of contents depends on the report document. The table of contents looks like Figure 4-21.

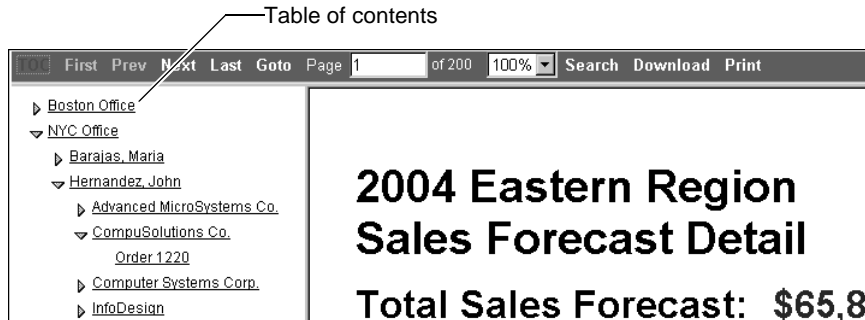


Figure 4-21 Table of contents

XMLDisplay is the only supported format for the table of contents.

Name <context root>\portal\activePortal\viewer\viewtoc.jsp

Parameters Table 4-43 lists and describes the parameters for the view TOC page. The view TOC page also uses the common URI parameters.

Table 4-43 TOC URI parameters

URI parameter	Description
componentID	The identifier of the report document component from which to retrieve Reportlet data. Specify either componentID, or componentName and componentValue.
connectionHandle	An identifier for temporary report documents only.
depth	The depth of the table of contents.
ID	The ID of the report document for which to create a table of contents.
name	The name of the report document for which to create a table of contents. This parameter is ignored if objectID is also specified.

Used by viewer\searchreport.jsp
viewer\viewframeset.jsp

Actuate Information Console JavaScript

This chapter contains the following topics:

- Actuate Information Console JavaScript overview
- Actuate Information Console JavaScript reference

Actuate Information Console JavaScript overview

This section describes the Actuate Information Console JavaScript files. Actuate Information Console JavaScript files provide functionality and dynamic content to Actuate Information Console web applications. Actuate Information Console JavaScript files reside in <context root>\portal\js.

Actuate Information Console JavaScript reference

Table 5-1 lists and describes the Actuate Information Console JavaScript files.

Table 5-1 Information Console JavaScript files

Name	Description
allscripts.js	Defines global variables, resources, and common methods such as deleteFile and viewActiveRequests
array.js	Contains functionality for handling arrays and array elements
browsertype.js	Determines the web browser in use and provides functionality appropriate to the browser, such as opening a file in a new window and capturing a keystroke event
calendarlayer.js	Provides calendar functionality for Information Console
converter.js	Provides character encoding
cookie.js	Provides cookie functionality, including reading, writing, and clearing browser cookies
drift.js	Adjusts layers and window display for Information Console
encoder.js	Contains the encode and unencode methods
help.js	Provides context-sensitive help functionality for Information Console
htmlselect.js	Provides methods for manipulating option controls
layer.js	Provides layer functionality, such as createLayer, deleteLayer, getWidth, showLayer
popupmenu.js	Defines the methods for manipulating pop-up menus
query.js	Provides the JavaScript components for parameter pages

Table 5-1 Information Console JavaScript files

Name	Description
report.js	Provides the JavaScript components for report viewing
requestsearch.js	Provides the JavaScript components for requesting searches
resize.js	Provides the JavaScript component for resizing a page for Information Console
saveas.js	Provides the JavaScript component for saving a file as another file or object
search.js	Provides search functionality for DHTML reports
searchtoolbar.js	Defines functionality for the Search toolbar buttons
skincustomization.js	Provides the JavaScript components for maintaining Actuate Information Console skins
strutscommon.js	Provides JavaScript components for using the Struts framework with Information Console
toctree.js	Builds and displays the DHTML report table of contents
viewer.js	Processes viewing parameters
viewframeset.js	Tracks the last page viewed for each report viaapplicationd for the current web browser session
viewframesetfuncs.js	Defines methods required to manage the viewer frameset such as resizing the frame, manipulating cookies, and retrieving values from a URI
viewnav.js	Displays the Actuate Information Console toolbar

Actuate Information Console servlets

This chapter contains the following topics:

- Information Console Java servlets overview
- Information Console Java servlets quick reference
- Information Console Java servlets reference

Information Console Java servlets overview

Java servlets extend web server functionality. Information Console uses Java servlets to manage binary content and to perform tasks such as uploading and downloading binary files. Actuate provides an abstract framework of servlets that provide common functionality to Information Console and Management Console. You cannot modify the Actuate Java servlets.

About the base servlet

All Actuate servlets derive from the base servlet:

```
com.actuate.reportcast.servlets.AcServlet
```

The base servlet has no URI parameters. It provides Actuate servlets with the functionality for performing the following tasks:

- Parse and validate parameters specified in Information Console URI directives.
- Create XML API structures based on Actuate Information Console requests.
- Submit XML streams to the Actuate SOAP API.
- Handle responses from the Actuate SOAP API, including detecting errors.
- Store constant session information, such as the name space and SOAP endpoint.
- Read from and write to cookies.
- Stream report data or errors to the web browser.

Invoking a servlet

Invoke servlets using the following syntax:

```
http://<application server>:<port>/<context root>/servlet  
/<servlet alias>
```

- application server is the name of the machine hosting the application server.
- port is the port on which the application server listens for requests.
- context root is the Information Console context root.
- servlet is a keyword indicating that a servlet follows.
- servlet alias is the name to which the servlet is mapped in the Information Console installation's web.xml file. A typical location for web.xml is C:\Program Files\Actuate11\iPortal\iportal\WEB-INF\web.xml.

Servlet names are case-sensitive. Do not modify the servlets, their names, or their mapping in web.xml.

Information Console Java servlets quick reference

Table 6-1 lists and describes the Information Console Java servlets.

Table 6-1 Actuate Information Console servlets

Information Console servlet	Description
DownloadFile servlet	Downloads a file from the Encyclopedia volume
DownloadSearchResult servlet	Downloads search results from an Actuate eReport in a specified format
ExecuteReport servlet	Submits a request to run a report
GetDynamicData servlet	Retrieves dynamic data, such as a chart from an Actuate e.report
GetReportData servlet	Displays the contents of components in an Actuate eReport
GetStaticData servlet	Retrieves static data such as images from an Actuate e.report
Interactive Viewer servlet	Displays a BIRT report document
ViewEmbeddedObject servlet	Displays embedded objects that are contained in an Actuate e.report
ViewPage servlet	Displays the contents of a specified report page

Information Console Java servlets reference

This section provides the detailed reference for Information Console servlets.

DownloadFile servlet

Downloads a file from the Encyclopedia volume.

Name com.actuate.reportcast.servlets.FileDownloadServlet

Invoke the DownloadFile servlet as:

http://<web server>:<port>/<context root>/servlet/DownloadFile

URI parameters Table 6-2 lists and describes the URI parameters for the DownloadFile servlet.

Table 6-2 Parameters for DownloadFile URI

URI parameter	Description
fileId	The unique identifier of an object, usually retrieved with the selectFilesFolders JSP tag.
name	The name of the object to download.
version	If name is specified, the version number of the object to view. If version is not specified, the latest version is retrieved.

DownloadSearchResult servlet

Downloads search results from an Actuate e.report file in a specified format.

Name com.actuate.reportcast.servlets.DownloadSearchResultServlet

Invoke the DownloadSearchResult servlet as:

http://<web server>:<port>/<context root>/servlet/DownloadSearchResult

URI parameters Table 6-3 lists and describes the URI parameters for the DownloadSearchResult servlet.

Table 6-3 Parameters for DownloadSearchResult URI

URI parameter	Description
enableColumnHeaders	If True, the first row of the output includes the column headers and the subsequent rows hold the data. If enableColumnHeaders is False, the output has no header row.
format	The format in which to download the search results. Values are: <ul style="list-style-type: none"> ■ Excel ■ CSV ■ TSV ■ ANALYSIS The default value is CSV.
hits	The point at which downloading of results ends.
id	The unique identifier of an object, usually retrieved with the selectFilesFolders JSP tag.
name	The name of the object the contents of which to search. If you provide a value for id, this parameter has no effect.

Table 6-3 Parameters for DownloadSearchResult URI

URI parameter	Description
outputName	<p>The name of the output file. The extension of the file name depends on the value of format. The following list shows the extensions for the format values:</p> <ul style="list-style-type: none">■ Excel. File extension is .xls.■ CSV. File extension is .csv.■ TSV. File extension is .txt. <p>If you provide a value for name, the name of the output file is the same as name and this parameter has no effect. If you provide a value for id, the default value for this parameter is output.</p>

To define the values to include in the output, you provide additional parameters to the servlet. The parameters are the fully scoped names of the controls to download, in one of the following formats:

- scope::controlName=* to select all values of a control.
- scope::controlName=value to select only the specified value for a control. This format supports the range of operators and wildcard characters that search criteria in the Search window of the DHTML Viewer support.
- scope::controlName:select=true to download all values of a control when another control is filtered with a value or wildcard string.

Example The following example downloads the data from two controls from a report. The search criteria filter the control ProductCodeControl in the OrderLineFrame frame to select only the values that begin with MR1 or MR3. All values for the control ProductDescControl in the same frame as a matching ProductCodeControl are also downloaded:

```
DownloadSearchResult?name=/eRDPro%20reports/SubReport.roi
&format=csv
&SubReportEx::OrderLineFrame::ProductCodeControl=MR1*,MR3*
&SubReportEx::OrderLineFrame::ProductDescControl:select=true
```

ExecuteReport servlet

Submits a request to iServer to run a report job. The execute report servlet is equivalent to do_executereport.jsp. This servlet supports executing spreadsheet reports. Excel does not support executing reports using do_executereport.jsp.

Name com.actuate.reportcast.servlets.ExecuteReportServlet

Invoke the ExecuteReport servlet as:

http://<web server>:<port>/<context root>/servlet/<report executable>

where the report executable is the ROX or SOX report file to execute.

URL parameters Table 6-4 lists and describes the parameters for the ExecuteReport servlet.

Table 6-4 Parameters for ExecuteReport URI

URI parameter	Description
__accessToGrant	The type of access to grant automatically to those roles that have permission to view the report.
__ageDays	Use with __ageHours to determine how long output objects exist before they are deleted. Use only if __archivePolicy is set to age. __ageDays can be any positive number.
__ageHours	Use with __ageDays to determine how long output objects exist before they are deleted. Use only if __archivePolicy is set to age. __ageHours can be any positive number.
__archiveBeforeDelete	Indicate whether to archive the output objects of the current request before deleting them, according to __archivePolicy's setting. Set to True to archive objects before deleting them. The default value is False. This parameter has no effect if __archivePolicy is set to folder.
__archivePolicy	The archive policy to implement for the objects created as output for the current request. Values are folder, age, and date. Set folder to use the archive policy that is already set for the folders to which the output is distributed. Set age to delete objects older than a specific time period. Set date to delete objects on a specific date.
__channels	Name of a channel to notify of this request. You can notify more than one channel.
__dateToDelete	The date on which to delete the output objects of the current request. Use only if __archivePolicy is set to date. __dateToDelete must be a date in a locale-specific format. The default format is mm/dd/yyyy.
__folder	The path name of the folder that contains the report executable.
__groups	The name of the group to notify of this request. You can notify more than one group.
__headline	A descriptive tag line for a report. Appears on the Channel Contents page. Use the character string %20 to represent spaces in the headline string.

Table 6-4 Parameters for ExecuteReport URI

URI parameter	Description
<code>__limit</code>	Indicate whether to limit the number of versions of the output files for the current request. Set <code>__limit</code> to limit to curtail the number of versions. Any other value means that the number of versions is unlimited.
<code>__limitNumber</code>	The number of versions to which to limit the output files for the current request. Use only if <code>__limit</code> is set to limit. <code>__limitNumber</code> can be any positive number.
<code>__outputName</code>	Specifies a name for the report output.
<code>__overwrite</code>	If True, overwrite any existing output. If False, do not overwrite existing output.
<code>__priority</code>	Specifies the job submission priority. Values are High, Medium, and Low.
<code>__priorityValue</code>	Specifies a number corresponding to the job submission priority.
<code>__recurringDay</code>	Specifies the scheduled recurring day on which to run the report job. Applies only to scheduled report jobs.
<code>__redirect</code>	Specifies a relative or absolute URL to go to after <code>do_executereport.jsp</code> submits the report. The default is <code>SubmittedJob_Status.jsp</code> .
<code>__saveInVolume</code>	Indicates whether to write the output document to the Encyclopedia volume. True saves the output in the Encyclopedia volume, applying the document archiving and file creation parameters. False does not save the output.
<code>serverURL</code>	Contains the URL that accesses the JSP engine, such as <code>http://Services:8080</code> .
<code>__timeToDelete</code>	Specifies a time at which to delete an archived report document. Applies only to scheduled report jobs.
<code>__users</code>	Contains the name of the user to notify of this request. You can notify more than one user.
<code>__versionName</code>	Contains a string value for the new version name of this report. The value can include a date/time expression enclosed in braces, {}, to ensure a unique version name.
<code>volume</code>	Contains a string value specifying the volume containing this report.
<code>__wait</code>	If "wait", Information Console waits for the report generation to be completed before displaying it. If "nowait", Information Console displays the first page right away even if the report is not completed.

GetDynamicData servlet

Retrieves dynamic data, such as charts from reports.

Name com.actuate.reportcast.servlets.GetDynamicDataServlet

Invoke the GetDynamicData servlet as:

http://<web server>:<port>/<context root>/servlet/GetDynamicData

URI parameters Table 6-5 lists and describes the URI parameters for the GetDynamicData servlet.

Table 6-5 Parameters for GetDynamicData URI

URI parameter	Description
componentID	The name of the stream that returns the dynamic object data, such as Image1.bmp.
connectionHandle	An identifier for temporary reports only.
ID	The unique identifier of an object, usually retrieved with the selectFilesFolders JSP tag.
name	The name of the object the contents of which to view.
scalingFactor	The scale at which to view the dynamic data, such as scalingFactor=100.
type	If the name option is specified, the type of the object to view, such as ROI.
version	If name is specified, the version number of the object to view. If version is not specified, the latest version is retrieved.

GetReportData servlet

Displays the contents of components in an Actuate e.report. Specify componentid of 0 (zero) to display the whole report. Specify component ID to display the data in a specific component. This servlet maps to the GetContent XML API.

Name com.actuate.reportcast.servlets.GetReportDataServlet

Invoke the GetReportData servlet as:

http://<web server>:<port>/<context root>/servlet/GetReportData

URI parameters Table 6-6 lists and describes the URI parameters for the GetReportData servlet.

Table 6-6 Parameters for GetReportData URI

URI parameter	Description
componentID	The identifier of the report component from which to retrieve Reportlet data. Specify either componentID, or componentName and componentValue.
componentName	The name of the report component from which to retrieve Reportlet data. The componentName is the name of the component assigned in the ROD file, such as NewReport::Frame1. If componentName is not specified, the componentID is used. Use in conjunction with componentValue. Specify either componentName and optionally componentValue, or componentID.
componentValue	A value identifying the specific instance of the report component from which to retrieve Reportlet data. The value is the result of evaluating a search expression. Use componentValue in conjunction with componentName. Specify either componentName and componentValue, or componentID. If unspecified, componentValue defaults to the first report component specified by componentName that the user has access to.
converterParam	Used internally by the converter framework for low-level parameters needed to generate desired output.
embSrvRequester	<p>The embSrvRequester parameter specifies a partial universal resource identifier (URI) to append to the base URI when inserting embedded objects.</p> <p>For example, adding "&embSrvRequester=/test/ViewEmbeddedObject" to the getReportData URI results in src elements in the generated HTML document with the following path:</p> <pre>SRC ="/test/ViewEmbeddedObject?..."</pre> <p>Without the parameter, the src element in the generated HTML document is the default:</p> <pre>SRC ="ViewEmbeddedObject?..."</pre>
encoding	Used by the view service. The character encoding as specified by the user.
format	<p>The file format in which to generate output. Values are:</p> <ul style="list-style-type: none"> ■ DHTML—a compressed DHTML format that uses cascading style sheets (CSS). This is the default format. ■ DHTMLLong—an uncompressed DHTML format. Use this format if your browsers do not support CSS. ■ DHTMLRaw—an uncompressed DHTML format without external JavaScript references, leaving the bare report.

(continues)

Table 6-6 Parameters for GetReportData URI (continued)

URI parameter	Description
format <i>(continued)</i>	<ul style="list-style-type: none"> ■ ExcelData—a format mainly used for tabular or listing reports. The appearance sometimes is not faithful to the original report or does not work well for complicated reports. Potential issues include undisplayed images and graphs, ignored background color of frames and flows, and only rough accuracy for component positioning. ■ ExcelDisplay—a format that resembles an Actuate report on an Excel spreadsheet as much as possible. <p>Reports in the ExcelData and ExcelDisplay formats are truncated if they exceed Maximum Number of Pages Convertible To Excel.</p> <ul style="list-style-type: none"> ■ PDF—report output in Adobe Acrobat-readable Portable Document Format. ■ RTF—report output in Rich Text Format. The report’s visual layout is similar to the DHTML Viewer layout. ■ RTFFullyEditable—report output in Rich Text Format, with more flexibility when manipulating output, such as the ability to move and delete several lines from a report at a time. Produces a larger RTF file than the RTF format. ■ XMLDisplay—a complete XML representation of the report. It contains information about attributes of controls. ■ XMLCompressedDisplay—the same as XML Display. The only difference is that it places all the common properties into a template. ■ XMLData—can only be generated if the report design has it specified. It is a property on the controls in the Designer for setting XML properties.
locale	The client’s locale in which to generate the report.
name	The name of the object the contents of which to view. Include a file extension in the name or set the file type using the type parameter.
ID	The unique identifier of an object, either the file id, for scheduled or save jobs, or an object id, for run jobs.
operation	The type of operation for the Actuate BIRT iServer to perform. Values are view or print. Default is view.
PDFQuality	The image quality in a PDF file. Use only if format is set to PDF. The value ranges from 100, for the lowest image quality but the smallest PDF file size, to 300, for the highest image quality but the largest PDF file size. Specify a larger charting heap size before using higher PDFQuality values. The default PDFQuality is an Actuate BIRT iServer advanced configuration value.

Table 6-6 Parameters for GetReportData URI (continued)

URI parameter	Description
reportletMaxHeight	Used by Reportlets. The maximum allowed height of Reportlets, measured in points.
scalingFactor	The scale at which to view the report, such as scalingFactor=100.
type	If the name parameter is specified, indicates the type of the object to view, such as ROI. GetReportData requires a type if the name parameter does not include a file extension.
userAgent	Used by the View service, such as Mozilla/4.0 (compatible; MSIE 5.01; Windows NT).
version	If name is specified, the version number of the object to view. If version is not specified, the latest version is retrieved.

GetStaticData servlet

Retrieves static data such as images from a report.

Name com.actuate.reportcast.servlets.GetStaticDataServlet

Invoke the GetStaticData servlet as:

http://<web server>:<port>/<context root>/servlet/GetStaticData

URI parameters Table 6-7 lists and describes the URI parameters for the GetStaticData servlet.

Table 6-7 Parameters for GetStaticData URI

URI parameter	Description
connectionHandle	An identifier for temporary reports only.
embed	Indicates whether an image is embedded or is external. 0 indicates an external image, and 1 indicates an embedded image.
ID	The unique identifier of an object, usually retrieved with the selectFilesFolders JSP tag.
name	The name of the object the contents of which to view.
streamName	The name of the stream which returns the static object data, such as Image1.bmp.
type	If the name option is specified, the type of the object to view, such as .ROI.
version	If name is specified, the version number of the object to view. If version is not specified, the latest version is retrieved.

Interactive Viewer servlet

Displays an Actuate BIRT report document with tools to affect the document and design files. The viewer has two modes, standard and interactive.

The Standard Viewer displays the report with toolbar options to save, print, show the TOC, and launch interactive mode, as shown in Figure 6-1.



Figure 6-1 Standard Viewer

The Interactive Viewer displays the report with toolbar options to navigate the report and provides context menus to edit and format report elements, as shown in Figure 6-2.



Figure 6-2 Interactive Viewer

The viewer supports rptdocument file formats.

Name com.actuate.iv.servlet.IVServlet

Invoke the Interactive Viewer servlet as:

http://<web server>:<port>/<context root>/iv

URI parameters Table 6-8 lists and describes the URI parameters for the Interactive Viewer servlet.

Table 6-8 Parameters for IV URI

URI parameter	Description
__bookmark	Name of the element of a report to display instead of the whole report file
__floatingfooter	Boolean value to add a margin under the footer

Table 6-8 Parameters for IV URI

URI parameter	Description
__format	<p>A format for the displayed report:</p> <ul style="list-style-type: none"> ■ pdf: Adobe PDF ■ xls: MS Excel ■ doc: MS Word ■ ppt: MS PowerPoint ■ ps: PostScript ■ html: HTML ■ flashchartsxml, flashgadgetsxml: used to display a fusion chart ■ reportlet: used together with __bookmark to show a particular part/element of the report
__from_page_range	The page range of a report to display
__from_page_style	<p>The page style to use for a report in pdf or ps formats:</p> <ul style="list-style-type: none"> ■ auto: The page size and content size remains the same. ■ actuateSize: Change the page size to fit the content. ■ fitToWholePage: Change the content size to fit the page size. <p>Used with the __format parameter</p>
__imageid	Name of the report file to display
__instanceid	Name of the report file to display
__launchiv	Boolean value that enables interactivity
__locale	Code for a locale
__page	A number for a page to render
__report	Name of the report file to display
__rtl	Name of the report file to display
repositoryType	The name of the object to download
serverURL	Contains the URL that accesses iServer, such as <code>http://ESL02835:8000</code>
userid	The user's identifier, required to log in to the Actuate BIRT iServer
volume	Contains a string value specifying the volume for this report

ViewEmbeddedObject servlet

The ViewEmbeddedObject servlet displays embedded objects that are contained in a report. The embedded objects can be static, such as external images or images

in an ROX; dynamic, such as charts or graphs; or embedded cascading style sheets (CSS). The ViewEmbeddedObject servlet is mapped to GetStaticData, GetDynamicData, or GetStyleSheet XML API depending upon the URI parameters. The report data generally embeds links pointing to the ViewEmbeddedObject servlet, including the appropriate URI parameters. The browser resolves these links automatically. Users can also view these objects directly by specifying the required parameters in the web browser.

Name com.actuate.reportcast.servlets.ViewEmbeddedObjectServlet

Invoke the ViewEmbeddedObject servlet as follows:

http://<web server>:<port>/<context root>/servlet/ViewEmbeddedObject

URI parameters Table 6-9 lists and describes the ViewEmbeddedObject servlet URI parameters.

Table 6-9 Parameters for ViewEmbeddedObject URI

URI parameter	Description
componentID	The unique object identifier for which to retrieve the data. Required for dynamic data.
embed	Indicates whether the object is embedded or external. 1 indicates an embedded object, and 0 indicates an external object.
objectID	The unique identifier of an object, usually retrieved with the selectFilesFolders JSP tag.
scalingFactor	The scale at which to view an object, such as scalingFactor=100 displays an object at full size.
streamName	Used only for static data. The name of the stream that returns the static object data, such as C:\Projects\Bmp\Image.bmp.

Example The following example retrieves a chart from a report. The chart's component ID, 198, and object ID, 17, were retrieved previously for use in this URI:

```
ViewEmbeddedObject?operation=GetDynamicData&ComponentID=198
&ObjectID=17&scalingFactor=100
```

ViewPage servlet

Displays the contents of a specified report page or range of pages in a web browser for paginated reports or all viewable contents if the report is unpaginated.

The ViewPage servlet also manages the display of embedded objects. Embedded objects include:

- Cascading style sheet (.css) files

- Static objects, such as images
- Dynamic objects, such as charts

The ViewPage servlet retrieves all the data for the report, including links to the view embedded objects servlet, which retrieves any embedded objects.

Name com.actuate.reportcast.servlets.ViewPageServlet

Invoke the ViewPage servlet as follows:

http://<web server>:<port>/<context root>/servlet/ViewPage

URI parameters Table 6-10 lists and describes the URI parameters for the ViewPage servlet.

Table 6-10 Parameters for ViewPage URI

URI parameter	Description
componentID	The identifier of the report component from which to retrieve Reportlet data. Specify either componentID, or componentName and componentValue.
componentName	The name of the report component from which to retrieve Reportlet data. The componentName is the name of the component assigned in the ROD file, such as NewReport::Frame1. If componentName is not specified, the componentID is used. Use in conjunction with componentValue. Specify either componentName and optionally componentValue, or componentID.
componentValue	A value identifying the specific instance of the report component from which to retrieve Reportlet data. The value is the result of evaluating a search expression. Use componentValue in conjunction with componentName. Specify either componentName and componentValue, or componentID. If unspecified, componentValue defaults to the first report component specified by componentName that the user has access to.
converterParam	Used internally by the converter framework for low level parameters needed to generate the desired output.
embSrvRequester	The embSrvRequester parameter specifies a partial universal resource identifier (URI) to append to the base URI when inserting embedded objects. For example, adding "&embSrvRequester=/test/ViewEmbeddedObject" to the viewPage URI results in src elements in the generated HTML document with the following path: SRC = "/test/ViewEmbeddedObject?..."

(continues)

Table 6-10 Parameters for ViewPage URI (continued)

URI parameter	Description
embSrvRequester (continued)	Without the parameter, the src element in the generated HTML document is the default: SRC ="ViewEmbeddedObject?..."
encoding	The character encoding specified by the user and required by the view service.
format	<p>The file format in which to generate output. Values are:</p> <ul style="list-style-type: none"> ■ DHTML—a compressed DHTML format that uses cascading style sheets (CSS). This is the default format. ■ DHTMLLong—an uncompressed DHTML format. Use this format if your browsers do not support CSS. ■ DHTMLRaw—an uncompressed DHTML format without external JavaScript references, leaving the bare report. ■ ExcelData—a format mainly used for tabular or listing reports. The appearance sometimes is not faithful to the original report or does not work well for complicated reports. Potential issues include undisplayed images and graphs, ignored background color of frames and flows, and only rough accuracy for component positioning. ■ ExcelDisplay—a format that appears as much like an Actuate report on an Excel spreadsheet as possible. <p>Reports in the ExcelData and ExcelDisplay formats are truncated if they exceed Maximum Number of Pages Convertible To Excel.</p> <ul style="list-style-type: none"> ■ PDF—report output in Adobe Acrobat-readable Portable Document Format. ■ Reportlet—the report is extracted as a part of the an HTML page. This means that the customary <Head> and <body> tags are not present. ■ RTF—report output in Rich Text Format. The report's visual layout is similar to the DHTML Viewer layout. ■ RTFFullyEditable—report output in Rich Text Format, with more flexibility when manipulating output, such as the ability to move and delete several lines from a report at a time. Produces a larger RTF file than the RTF format. ■ XMLDisplay—a complete XML representation of the report. It contains information about attributes of controls. ■ XMLCompressedDisplay—the same as XML Display. The only difference is that it places all the common properties into a template.
locale	The client's locale in which to generate the report.

Table 6-10 Parameters for ViewPage URI (continued)

URI parameter	Description
mode	The page to which to navigate. Values are first, last, previous, and next. If mode is specified, page is ignored.
name	The name of the object the contents of which to view.
objectID	The ID of an object, usually retrieved with the selectFilesFolders JSP tag.
operation	The type of operation for the Actuate BIRT iServer to perform. Values are view or print. Default is view.
page	The object's page to display. If mode is specified, page is ignored.
pageHeight	The printed page height in twips. Used only when format=PDF and splitOversizePages=1. This page height overrides the report's page height and any page splitting rules in the report.
pageWidth	The printed page width in twips. Used only when format=PDF and splitOversizePages=1. This page width overrides the report's page width and any page splitting rules in the report.
PDFQuality	The image quality in a PDF file. Use only if format is set to PDF. The value ranges from 100, for the lowest image quality but the smallest PDF file size, to 300, for the highest image quality but the largest PDF file size. Specify a larger charting heap size before using higher PDFQuality values. The default PDFQuality is an Actuate BIRT iServer advanced configuration value.
range	The range of pages to retrieve from the object and display in a specified output file format. Separate pages and page ranges with commas, such as 1-3, 15, 21-25.
reportletMaxHeight	Specifies the maximum height for Reportlets, measured in points.
scalingFactor	The scale at which to view an object, such as scalingFactor=100.
searchList	<p>The list of name-value pairs that uniquely identify a page or pages containing components on which to search. The format is:</p> <pre>&<component name>=<value> [&<component name> = <value>] ...</pre> <p>where <component name> is the fully qualified name of the component on which the search condition is based. Do not enter the searchList parameter name. Enter only the component/value pairs. For example, titleframe::txname:include=*.searchList finds the first matching component in the report because it is a component-based search and not page-based. This means that it does not always return the page containing both the name and value, but the first section or component that matches the search criteria.</p>

(continues)

Table 6-10 Parameters for ViewPage URI (continued)

URI parameter	Description
splitOversizePages	Split report pages to print on multiple sheets. Values are 0, the default, to not split and 1 to split. SplitOversizePages is ignored unless format=PDF. Use if the report's page size is larger than the printer's sheet size. For information about setting the output page size, see pageHeight and pageWidth in this section.
type	If the name option is specified, this value is the type of the object to view, such as ROI.
userAgent	Used by the view service, such as Mozilla/4.0 (compatible; MSIE 5.01; Windows NT).
version	If name is specified, the version number of the object to view. If version is not specified, the latest version is retrieved.

Example The following example displays page 1 of the report Msbargph.roi in DHTML format:

```
http://mycorp:8700/iportal/servlet/ViewPage?name=msbargph
&type=ROI&format=DHTML&page=1
```

Actuate Information Console custom tags

This chapter contains the following topics:

- Information Console custom tag overview
- Information Console custom tags quick reference
- Information Console custom tags reference

Information Console custom tag overview

This chapter provides reference information about Information Console tag libraries and their custom tags. Custom tags are JSP language elements that you define to encapsulate frequent tasks. A tag library defines a set of related custom tags and contains the objects that implement the tags.

The Information Console tag libraries reside in <context root>\WEB-INF. The tag libraries define the XML tags and attributes that the Information Console pages use. Examine individual pages to determine the tag libraries that they use. For example, viewdefault.jsp uses the internationalization, common, and users tag libraries, as shown in the following example:

```
<%-- DECLARE ANY RESOURCE BUNDLES USED IN THIS PAGE --%>
<%@ taglib uri="/i18n" prefix="i18n"%>
<%@ taglib uri="/common" prefix="common" %>
<%@ taglib uri="/users" prefix="users" %>
```

You declare that a page uses tags by including the taglib directive in the page before you use any custom tag. The uri attribute refers to a URI (Uniform Resource Identifier) that uniquely identifies the tag library descriptor (TLD). A TLD file is an XML document that describes a tag library. The prefix attribute defines the prefix that distinguishes tags defined by a given tag library from those provided by other tag libraries. The prefix can differ for each use of the taglib statement, but every prefix must be unique within a page.

Information Console custom tag names are case sensitive.

Information Console custom tags quick reference

This section provides two quick reference lists related to Information Console custom tags:

- Information Console custom tag libraries
- Information Console custom tags

Information Console custom tag libraries

Table 7-1 lists and describes the Information Console custom tag libraries.

Table 7-1 Actuate Information Console tag libraries

Tag library	Description
actabpanel	Provides tags for creating tabbed pages
common	Provides tags storing and iterating through data

Table 7-1 Actuate Information Console tag libraries

Tag library	Description
filefolders	Tags for managing files and folders
i18n	Provides tags for internationalization
login	Provides tags for login operation
reportlet	Provides tags to retrieve Reportlet data
viewer	Provides tags for viewing report documents

Information Console custom tags

Information Console uses Jakarta Struts custom tags and Actuate custom tags. Actuate recommends that customized Information Console web applications use Jakarta Struts custom tags and only the Actuate custom tags shown in Table 7-2.

Table 7-2 Actuate Information Console custom tags

Tag library	Tag	Description
actabpanel	content	Specifies the page to display when the user or URL selects the associated tab
actabpanel	tab	Specifies the label on a page's tab and its key
actabpanel	tabBegin	Specifies any HTML or JSP code to apply before defining any tabs
actabpanel	tabEnd	Specifies any HTML or JSP code to apply after defining all tabs
actabpanel	tabMiddle	Specifies any HTML or JSP code to apply to each unselected tab
actabpanel	tabMiddleSelected	Specifies any HTML or JSP code to apply to the selected tab
actabpanel	tabPanel	Contains all tags defining page tabs
actabpanel	tabSeparator	Specifies any HTML or JSP code to apply between each adjacent pair of tabs
common	iterator	Iterates through a collection of data
common	string	Holds a single string of data
common	stringList	Holds an array of strings
filefolders	copyFileFolder	Copies files and folders

(continues)

Table 7-2 Actuate Information Console custom tags (continued)

Tag library	Tag	Description
i18n	bundle	Wraps org.apache.taglibs.i18n.BundleTag
i18n	formatDate	Formats a Date value using a locale
i18n	message	Allows the usage of a resource bundle to internationalize content
login	login	Performs the login operation
reportlet	getReportlet	Displays a page or other subset of an Actuate eReport as a Reportlet
reportlet	getReportletData	Displays an Actuate eReportlet
viewer	component	Specifies the component
viewer	componentIdentifier	Specifies the component identifier for SmartSearch
viewer	componentIdentifierList	Specifies a list of component identifiers for SmartSearch
viewer	componentList	Specifies the component list identified by the name attribute
viewer	getFormats	Fetches the formats supported by view server
viewer	getPageCount	Fetches the page count of a report
viewer	getTOC	Gets the report's table of contents data from the Actuate BIRT iServer
viewer	searchReport	Fetches matching data

Information Console custom tags reference

This section provides the detailed reference for Information Console custom tags.

bundle

Establishes the ResourceBundle to use for other i18n tags in the JSP. It also determines the most appropriate locale to use based on browser settings if a locale is not provided. It overrides the doEndTag() method and sets the ChangeResponseLocale feature to False. This tag must be placed in a JSP before any other i18n tags. This tag wraps the org.apache.taglibs.i18n.BundleTag.

- Library** i18n
- Tag class** com.actuate.reportcast.tags.common.BundleTag
- Attributes** Table 7-3 lists and describes the attributes for bundle.

Table 7-3 Attributes for bundle

Attribute	Required	Description
baseName	Yes	Used along with the locale to locate the desired ResourceBundle
id	No	Variable ID for use with standard <code>jsp:getProperty</code> tag and as an attribute to other tags in this tag library
locale	No	Current user's locale, such as en_US, from <code><context root>\WEB-INF\localemap.xml</code>
localeAttribute	No	Name of an attribute whose value is the user's preferred locale

- Variables** Table 7-4 describes the variable for bundle.

Table 7-4 Variable for bundle

Variable	Description
id	Allows other tags or scriptlets to access the ResourceBundle defined by this tag. This is useful for allowing multiple bundle declarations per page or for creating localization debug pages by listing all key and value pairs in a bundle.

- Example** The following example defines a bundle using browser preference to determine locale:

```
<i18n:bundle baseName="com.mycorp.taglibs.i18n.test"/>
```

The next example defines a bundle using browser preference to determine locale, and declaring the scripting variable bundle:

```
<i18n:bundle baseName="com.mycorp.taglibs.i18n.test" id="bundle"/>
```

The next example defines a bundle using a scriptlet variable to specify the locale:

```
<i18n:bundle baseName="com.mycorp.taglibs.i18n.test"
  locale="<%= localeVar %>"/>
```

- Used in** `<context root>\errors\error.jsp`
`<context root>\errors\pagenotfound.jsp`
`<context root>\viewer\closewindow.jsp`
`<context root>\viewer\print.jsp`
`<context root>\viewer\saveas.jsp`

```
<context root>\viewer\searchreport.jsp
<context root>\viewer\viewreport.jsp
<context root>\viewer\waitforexecution.jsp
```

component

Specifies the report component on which to search.

- Library** viewer
- Tag class** com.actuate.reportcast.tags.viewer.ComponentTag
- Attributes** Table 7-5 lists and describes the attributes for component.

Table 7-5 Attributes for component

Attribute	Required	Description
name	No	The report component's name
objectID	No	The report component's identifier
value	Yes	The report component's value

- Used in** <context root>\viewer\searchframe.jsp
- Example** The following example specifies a search for manager names that begin with the letter B:

```
<viewer:component name="Frame1::ManagerName" value="B*" />
```

componentIdentifier

Provides the component identifier for searches.

- Library** viewer
- Tag class** com.actuate.reportcast.tags.viewer.ComponentIdentifierTag
- Attributes** Table 7-6 lists and describes the attributes for componentIdentifier.

Table 7-6 Attributes for componentIdentifier

Attribute	Required	Description
id	No	The report component's ID
name	No	The report component's name

- Used in** <context root>\viewer\searchframe.jsp

Example The following example removes leading and trailing spaces from the sParamId value and sets the component identifier to the value:

```
<viewer:componentIdentifier id="<%= sParamId.trim() %>" />
```

componentIdentifierList

Provides a list of component identifiers for searches.

Library viewer

Tag class com.actuate.reportcast.tags.viewer.ComponentIdentifierListTag

Attributes Table 7-7 describes the attribute for componentIdentifierList.

Table 7-7 Attribute for componentIdentifierList

Attribute	Required	Description
name	Yes	The component list name

Used in <context root>\viewer\searchframe.jsp

Example The following example sets the name of the list of identifiers to SelectList:

```
<viewer:componentIdentifierList name="SelectList">
```

componentList

Provides an array of components on which to search. This tag is used for specifying the component list that is identified by the name attribute. It holds an array of components.

Library viewer

Tag class com.actuate.reportcast.tags.viewer.ComponentListTag

Attributes Table 7-8 describes the attribute for componentList.

Table 7-8 Attribute for componentList

Attribute	Required	Description
name	Yes	The element to create for searching the report

Used in <context root>\viewer\searchframe.jsp

Example The following example sets the name of the array of components to SearchByNameList:

```
<viewer:componentList name="SearchByNameList" />
```

content

Specifies the content of a page beneath a tab. Include HTML or JSP code in the body of the content tag or use the page attribute to include another JSP file as the content.

- Library** actabpanel
- Tag class** com.actuate.activeportal.tags.tabpanel.Content
- Attributes** Table 7-9 describes the attribute for content.

Table 7-9 Attribute for content		
Attribute	Required	Description
page	No	Specifies a file containing the JSP code to use as the content of the page associated with the current tab

If you do not include a page attribute, any HTML or JSP code in the tag’s body becomes the definition of the page.

- Used in** <context root>\private\jobs\selectjobscontent.jsp
<context root>\private\newrequest\newrequestpage.jsp
<context root>\private\options\optionspage.jsp
<context root>\private\query\createpage.jsp
<context root>\private\query\runpage.jsp

Examples The following example uses the page attribute to specify using the code in saveas.jsp as the Save As tab’s page content:

```
<ui:tab><bean:message key="TAB_SAVE_AS"/></ui:tab>
<ui:content page="saveas.jsp"/>
```

The following example uses the tag’s body to specify the About tab’s content page as the result of the JSP include directive:

```
<ui:tab key="about" unselected="class=\"lnkTab\"">
  <bean:message key="TAB_ABOUT"/>
  <ui:content>
    <%@ include file="about.jsp" %>
  </ui:content>
</ui:tab>
```

copyFileFolder

Copies files and folders from one location to another.

Library filesfolders

Tag class com.actuate.reportcast.tags.filesfolders.CopyFileFolderTag

Attributes Table 7-10 lists and describes the attributes for copyFileFolder.

Table 7-10 Attributes for copyFileFolder

Attribute	Required	Description
appendFileName	No	Boolean value. If True, add newName to the targetPath. Default is True.
authID	Yes	Unique authentication ID assigned to the user after successful login.
createNewVersion	No	Boolean value. If True, create a new version of the file or folder. Default is True.
latestVersionOnly	No	Boolean value. If True, only the latest version is to be copied. Default is True.
locale	No	Current user's locale, such as en_US, from <Context root>\WEB-INF\localemap.xml.
maxVersions	No	Number of versions to copy.
newName	No	New name for copied item.
serverURL	Yes	URL that accesses the BIRT iServer, such as http://Services:8000.
targetPath	Yes	Directory path to which to copy.
timeZone	No	Current user's time zone from <Context root>\Web-inf\TimeZone.xml.
volume	Yes	BIRT iServer volume to copy from.
workingFolderID	No	Unique ID for the source folder.
workingFolderName	No	Name of the source folder.

formatDate

Formats a Date value using a locale. A style or a pattern such as 'YYYY MMM ddd' is specified. If the value is null then the default text is used. If no locale is specified then the parent locale tag is used. If no parent locale tag exists then the locale is taken from the current request. If still no locale is found then the current JVM locale is used.

Library i18n

getFormats

Tag class org.apache.taglibs.i18n.FormatDateTag

Attributes Table 7-11 lists and describes the attributes for formatDate.

Table 7-11 Attributes for formatDate

Attribute	Required	Description
defaultText	No	Default value.
locale	No	Current user's locale, such as en_US, as in <context root>\Web-inf\localemap.xml.
pattern	No	Date formatting string. Do not use with style.
style	No	Short, medium, long, or full. Do not use with pattern.
value	No	Date value.

Used in <context root>\private\channels\channelnoticelistcontent.jsp
<context root>\private\filesfolders\filedetailcontent.jsp
<context root>\private\jobs\completedjob.jsp
<context root>\private\jobs\getjobdetailscontent.jsp
<context root>\private\jobs\pendingjob.jsp
<context root>\private\jobs\runningjob.jsp
<context root>\private\jobs\scheduledjob.jsp

getFormats

Returns the report output formats supported by the Actuate BIRT iServer.

Library viewer

Tag class com.actuate.reportcast.tags.viewer.GetFormatsTag

Attributes Table 7-12 lists and describes the attributes for getFormats.

Table 7-12 Attributes for getFormats

Attribute	Required	Description
authID	Yes	The unique authentication identifier returned by Actuate BIRT iServer on successful login
connection Handle	No	The view server connection handle from the Actuate BIRT iServer
formatType	No	0, 1, or 2. 0 indicates all formats, 1 indicates view, and 2 indicates search
id	No	The object's identifier

Table 7-12 Attributes for getFormats

Attribute	Required	Description
locale	Yes	The desired locale, such as en_US, from <context root>\Web-inf\localemap.xml
name	No	The object's name
serverURL	Yes	The host and port for the Actuate BIRT iServer machine, such as http://Services:9000
timeZone	Yes	The desired time zone from <context root>\Web-inf\TimeZone.xml
type	No	The object's type, such as ROI
version	No	The object's version number
volume	Yes	The Encyclopedia volume to use

Variables llformats: the linked list of formats supported by the Actuate BIRT iServer

Used in <context root>\viewer\searchframe.jsp

Example The following example gets the view server formats for version four of the Detail.roi file:

```
<viewer:getFormats name="Costs.roi" version="4" formatType=2 />
```

getPageCount

Returns the total number of pages in a report.

Library viewer

Tag class com.actuate.reportcast.tags.viewer.GetPageCountTag

Attributes Table 7-13 lists and describes the attributes for getPageCount.

Table 7-13 Attributes for getPageCount

Attribute	Required	Description
authID	Yes	The unique authentication identifier returned by Actuate BIRT iServer on successful login
connection Handle	No	The connection handle from the Actuate BIRT iServer
id	No	The object's identifier
locale	Yes	The desired locale, such as en_US, as in <context root>\Web-inf\localemap.xml

(continues)

Table 7-13 Attributes for getPageCount (continued)

Attribute	Required	Description
name	No	The object's name
serverURL	Yes	The host and port for the Actuate BIRT iServer machine, such as http://Services:9000
type	No	The object's type, such as ROI
version	No	The object's version number
volume	Yes	The Encyclopedia volume to use

Used in <context root>\viewer\print.jsp
 <context root>\viewer\saveas.jsp
 <context root>\viewer\validatefile.jsp
 <context root>\viewer\viewnavigation.jsp

Example The following example returns the number of pages in the report whose unique identifier is 4:

```
<viewer:getPageCount objectID="4" />
```

getReportlet

Used to display a Reportlet. This tag is similar to the getReportletData tag but is used to get the Reportlet as a page. This tag sends the request to Actuate BIRT iServer to retrieve the page, then parses the response and writes the DHTML Reportlet to the JSP output stream. You access the height and width of the Reportlet by calling the getHeight() and getWidth() methods respectively.

Library reportlet

Tag class com.actuate.reportcast.tags.reportlet.GetReportletTag

Attributes Table 7-14 lists and describes the attributes for getReportlet.

Table 7-14 Attributes for getReportlet

Attribute	Required	Description
authID	Yes	The unique authentication identifier returned by Actuate BIRT iServer on successful login.
componentID	No	The identifier of the report component from which to retrieve Reportlet data. Specify either componentID, or componentName and componentValue.
component Name	No	The name of the report component from which to retrieve Reportlet data. The componentName

Table 7-14 Attributes for getReportlet (continued)

Attribute	Required	Description
component Name (continued)	No (continued)	is the name of the component assigned in the ROD file, such as NewReport::Frame1. If componentName is not specified, the componentID is used. Use in conjunction with componentValue. Specify either componentName and optionally componentValue, or componentID.
component Value	No	A value identifying the specific instance of the report component from which to retrieve Reportlet data. The value is the result of evaluating a search expression. Use componentValue in conjunction with componentName. Specify either componentName and componentValue, or componentID. If unspecified, componentValue defaults to the first report component specified by componentName that the user has access to.
connection Handle	No	The connection handle from the Actuate BIRT iServer.
customInput Para	No	Parameters to a custom converter for a new output type.
embeddedObj Path	No	The path name of the servlet which retrieves embedded objects such as images for the Reportlet.
fileId	No	The report file's unique identifier.
id	No	The object's unique identifier.
locale	No	The Reportlet locale, such as en_US, as in <context root>\Web-inf\localemap.xml.
name	No	The Reportlet source's full Encyclopedia volume path name.
operation	No	Operation to perform, view or print.
page	No	The single page to display as a Reportlet.
range	No	A range of pages to include. Separate pages and page ranges with commas, such as: 1-3, 15, 21-25.
reportletMax height	No	The Reportlet's maximum height in points.

(continues)

Table 7-14 Attributes for getReportlet (continued)

Attribute	Required	Description
scalingFactor	No	Less than 100 reduces and more than 100 enlarges the Reportlet.
searchCriteria	No	Search criteria to use.
serverURL	Yes	The URL of the server on which the Reportlet's source resides.
timeZone	Yes	The Reportlet server's time zone as in <context root>\Web-inf\TimeZone.xml.
useragent	No	The user's browser, such as IE/5.5.
version	No	The version number for the Reportlet source.
viewMode	No	Additional viewing parameters.
volume	Yes	The Encyclopedia volume on which the Reportlet's source resides.

getReportletData

Retrieves data to display in an Actuate eReport Reportlet. This tag sends the request to the Actuate BIRT iServer to get the Reportlet. Then it parses the response and writes the DHTML Reportlet to the JSP output stream. You access the height and width of the Reportlet by calling the getHeight() and getWidth() methods respectively.

Library reportlet

Tag class com.actuate.reportcast.tags.reportlet.GetReportletDataTag

Attributes Table 7-15 lists and describes the attributes for getReportletData.

Table 7-15 Attributes for getReportletData

Attribute	Required	Description
authID	Yes	The unique authentication identifier returned by Actuate BIRT iServer on successful login.
componentID	No	The identifier of the report component from which to retrieve Reportlet data. Specify either componentID, or componentName and componentValue.
component Name	No	The name of the report component from which to retrieve Reportlet data. The componentName is the name of the component assigned in the ROD file, such as NewReport::Frame1. If

Table 7-15 Attributes for getReportletData

Attribute	Required	Description
component Name (continued)	No (continued)	componentName is not specified, the componentID is used. Use in conjunction with componentValue. Specify either componentName and optionally componentValue, or componentID.
component Value	No	A value identifying the specific instance of the report component from which to retrieve Reportlet data. The value is the result of evaluating a search expression. Use componentValue in conjunction with componentName. Specify either componentName and componentValue, or componentID. If unspecified, componentValue defaults to the first report component specified by componentName that the user has access to.
connection Handle	No	The connection handle from the Actuate BIRT iServer.
embeddedObj Path	No	The path name of the servlet which retrieves embedded objects such as images for the Reportlet.
fileId	No	The report's unique identifier.
hyperlink RedirectPath	No	The path name of the servlet which handles Reportlet URLs. For example, iportal/servlet/GenericRedirector. The URLs are generated in the DHTML output, and hyperlinkRedirectPath is the base path for the generated URL.
locale	No	The Reportlet locale.
name	No	Reportlet source's full Encyclopedia path name.
objectID	No	The getReportletData tag's identifier.
reportletMax height	No	The Reportlet's maximum height in points.
serverURL	Yes	The URL of the server on which the Reportlet's source resides.
timeZone	Yes	The Reportlet server's time zone.
useragent	No	The user's browser, such as IE/5.5.
version	No	The version number for the Reportlet source.
volume	Yes	The Encyclopedia volume on which the Reportlet's source resides.

Example The following code retrieves a Reportlet from the Stock Comparison report in the Sample Application's Customers folder. The Reportlet is retrieved from the ROD component NewReportApp::Frame2.

```
<reportlet:getReportletData authID="<%= sAuthID %>"
  volume="<%= sVolume %>"
  serverURL="<%= sServerURL %>"
  locale="<%= acLocale %>"
  timeZone="<%= tzTimeZone %>"
  objectID="123"
  name="/Customers/StockComparison.roi"
  componentName="NewReportApp::Frame2"
  componentValue="1"
  reportletMaxheight="300"
  version="1"
  embeddedObjPath=" ../servlet/ViewEmbeddedObject?operation=" >
</reportlet:getReportletData>
```

getTOC

Obtains the table of contents for a report from the server. It fetches the data in XML form, which is converted for display in the browser. The report is identified by its ID or name. Either the report ID or name attribute must be set.

Library viewer

Tag class com.actuate.reportcast.tags.viewer.GetTOCTag

Attributes Table 7-16 lists and describes the attributes for getTOC.

Table 7-16 Attributes for getTOC		
Attribute	Required	Description
authID	Yes	The unique authentication identifier returned by Actuate BIRT iServer on successful login.
connection Handle	No	The view server connection handle from the Actuate BIRT iServer.
depth	No	The depth of the table of contents.
format	No	The table of contents' format. XMLDisplay is the supported format.
id	No	The object's unique identifier.
locale	Yes	The report's locale, such as en_US, as in <context root>\Web-inf\localemap.xml.
name	No	The name of the object for which a table of contents is being built.

Table 7-16 Attributes for getTOC

Attribute	Required	Description
nodeId	No	The identifier specifying the starting point of the table of contents.
serverURL	Yes	The URL of the server on which the report resides, such as <code>http://Services:9000</code> .
timeZone	Yes	The report's time zone as in <code><context root>\Web-inf\TimeZone.xml</code> .
type	No	The type of object, such as ROI.
userAgent	No	The user's browser, such as IE/5.5.
version	No	The report's version. The latest version is used if no version is given.
volume	Yes	The report's Encyclopedia volume.

Used in `<context root>\viewer\viewtoc.jsp`

Example The following example creates the table of contents for the Detail report:

```
<viewer:getToc name="Details.roi" type="roi" format="XMLDisplay"/>
```

iterator

Supports iterating through the contents of lists and retrieving specified parameters. The iterator tag populates its invoking JSP with the contents of the list. Information Console and Management Console JSPs make extensive use of the iterator tag to process lists.

Library common

Tag class `com.actuate.reportcast.tags.common.IteratorTag`

Attributes Table 7-17 lists and describes the attributes for iterator.

Table 7-17 Attributes for iterator

Attribute	Required	Description
collection	No	The collection through which to iterate.
content	No	The unique identifier of the results through which to iterate.
name	Yes	The unique identifier to use while retrieving values from this iterator.
type	Yes	Fully qualified name representing the type of listed objects through which to iterate.

login

Variables isLastRow: False while iteration is in progress. True when iteration reaches the last row.

Used in <context root>\viewer\viewdefault.jsp

Example The following example iterates through a list of job notices:

```
<cmn:iterator name="igjn"
  type="com.actuate.reportcast.dstruct.JobNotice"
  content="gjn">
```

login

Establishes the connection to the Actuate BIRT iServer. On successful connection, Actuate BIRT iServer returns the authentication ID used during the user's session to validate credentials and check access permissions. Actuate BIRT iServer returns the home folder and start folder as well.

Library login

Tag class com.actuate.reportcast.tags.common.LoginTag

Attributes Table 7-18 lists and describes the attributes for login.

Table 7-18 Attributes for login

Attribute	Required	Description
disableBasic Authentication	No	Boolean value. If True, disable basic authentication.
force	No	Boolean value. True to force a login, False to display the login page. The default is False. For example, when switching between Encyclopedia volumes and using APSE, set force=true to force the Information Console Login module to call APSE to perform the login operation. This prevents the login page from appearing unnecessarily.
id	Yes	Unique identifier for the object.
locale	No	The locale to display.
password	No	The user's password.
serverURL	No	The host and port of the Actuate BIRT iServer to which the user wants to connect.
timeZone	No	The time zone to display.

Table 7-18 Attributes for login

Attribute	Required	Description
userID	Yes	The user's identifier, required to log in to the Actuate BIRT iServer.
volume	No	The name of the volume to which the user wants to connect. If volume is not specified, the login tag checks whether the variable volume_default is set in the web.xml file.

Variables Table 7-19 lists and describes the variables for login.

Table 7-19 Variables for login

Variable	Description
homeFolder	The user's home folder
startFolder	The folder that the user sees upon successful login
authID	The authentication ID returned by Actuate BIRT iServer upon successful login

Used in <context root>\authenticate.jsp

Example The following example logs in the user jaguilar to the volume sales on the Actuate BIRT iServer marcom with the password secret:

```
<actu:login clusterURL="http://marcom:8900/" userID="jaguilar"
  password="secret" volume="sales" />
```

message

Implements a body tag allowing the usage of a resource bundle to internationalize content in a web page. The key attribute is required, and is used to look up content in the resource bundle. The args attribute is optional, and if present, provides items to pass to a MessageFormat. The bundle tag must be used first in order to ensure that the proper bundle is loaded.

Library i18n

Tag class org.apache.taglibs.i18n.MessageTag

Attributes Table 7-20 lists and describes the attributes for message.

Table 7-20 Attributes for message

Attribute	Required	Description
args	No	An array of arguments for use with <code>java.text.MessageFormat</code> when formatting the display text.
bundle	No	Object reference to the <code>ResourceBundle</code> in which the key can be found.
bundleRef	No	Name of an attribute that contains a resource bundle.
key	Yes	Key to use when retrieving the display message format from the <code>ResourceBundle</code> .

Used in <context root>\errors\pagenotfound.jsp

Variables Table 7-21 describes the variable for message.

Table 7-21 Variable for message

Variable	Description
id	id allows other tags or scriptlets to access the String created by this tag. If id is specified the String is not printed by this tag, just stored into the id.

Example The following example displays a plain message using the default (first defined) bundle:

```
<i18n:message key="column1.header"/>
```

The next example displays a plain message using a specified bundle. In this example the default bundle is `bundle1` because it is defined first:

```
<i18n:bundle baseName="com.mycorp.taglibs.i18n.i18n-test"
  id="bundle1"/> <!-- the default -->

<i18n:bundle baseName="com.mycorp.taglibs.i18n.i18n-test2"
  id="bundle2"/> <!-- the alternate -->

<i18n:message key="column1.header" bundle="<%= bundle2 %>" />
```

searchReport

Fetches data corresponding to the specified search conditions in a report.

Library viewer

Tag class com.actuate.reportcast.tags.viewer.SearchReportTag

Attributes Table 7-22 lists and describes the attributes for searchReport.

Table 7-22 Attributes for searchReport

Attribute	Required	Description
authID	Yes	The unique authentication identifier returned by Actuate BIRT iServer on successful login.
connection Handle	No	The connection handle from the Actuate BIRT iServer.
enableColumn Headers	No	Boolean value. If True, enable column headers.
format	Yes	The search results format. Possible values are: <ul style="list-style-type: none"> ■ DISPLAY for DHTML display ■ ANALYSIS for e.Analysis output ■ CSV for comma separated values ■ TSV for tab separated values
frameset	No	Frameset in which to search.
hits	No	Number of matches to find.
id	No	The object's identifier.
locale	Yes	The report's locale.
name	No	The name of the object.
serverURL	Yes	The URL of the server on which the report resides, such as http://Services:9000.
startingPoint	No	The starting point for the result set.
timeZone	Yes	The report's time zone as in <context root>\Web-inf\TimeZone.xml.
type	No	The type of object, such as ROI.
useQuote Delimiter	No	Boolean value. If True, delimit values with quotes.
userAgent	No	The user's browser, such as IE/5.5.
version	No	The report version to search.
volume	Yes	The Encyclopedia volume to search.

Used in <context root>\viewer\searchframe.jsp

selectUsers

Retrieves detailed user information for the current user. The list of users to be selected can be specified by means of a filter condition or a fetch handle.

Library users

Tag class com.actuate.reportcast.tags.users.SelectUsersTag

Attributes Table 7-23 lists and describes the attributes for selectUsers.

Table 7-23 Attributes for selectUsers

Attribute	Required	Description
authID	Yes	Unique authentication ID assigned to the user after successful login.
countLimit	No	Maximum number of users to retrieve.
fetchAction	No	Sort order for the returned list. Use True, the default value, for ascending order and False for descending order.
fetchHandle	No	Optional handle obtained from a previous list retrieval request. If fetchHandle is passed then filter and status are ignored.
fetchSize	No	Number of records to retrieve.
filter	No	Filter condition to apply to the selected jobs. The filter applies only to the job names. The default is to select all jobs.
id	Yes	Unique identifier of the object.
locale	No	Current user's locale, such as en_US, as in <context root>\Web-inf\localemap.xml.
serverURL	Yes	URL that accesses the Actuate BIRT iServer, such as http://Services:9000.
status	No	User status to search for.
timeZone	No	Current user's time zone as in <context root>\Web-inf\TimeZone.xml.
volume	Yes	Encyclopedia volume that contains the users.

Used in <context root>\viewer\viewdefault.jsp

string

Holds a single string of data.

Library common

Tag class com.actuate.reportcast.tags.common.StringTag

Attributes Table 7-24 lists and describes the attributes for string.

Table 7-24 Attributes for string

Attribute	Required	Description
name	No	String name
value	Yes	String contents

Used in <context root>\viewer\searchframe.jsp

stringList

Holds an array of strings.

Library common

Tag class com.actuate.reportcast.tags.common.StringListTag

Attributes Table 7-25 lists and describes the attributes for stringList.

Table 7-25 Attributes for stringList

Attribute	Required	Description
collection	No	Array of strings
items	No	Length of array
name	Yes	Array name
property	No	Property of the array

Used in <context root>\viewer\searchframe.jsp
<context root>\viewer\viewdefault.jsp

tab

Defines the label and key for a tab in a tab panel. URIs specifying the key cause selection of the tab and display of the page associated with the tab.

tabBegin

Library actabpanel

Tag class com.actuate.activeportal.tags.tabpanel.Tab

Attributes Table 7-26 lists and describes the attributes for tab.

Table 7-26 Attributes for tab

Attribute	Required	Description
key	No	Specifies the identification key for this tab. If not set, the default is 0, 1, 2 and so on. Use with the selectedTab attribute of the tabPanel tag.
selected	No	Specifies the label on the tab while the tab is selected.
unselected	No	Specifies the label on the tab while the tab is not selected.

Used in <context root>\private\common\errors\error.jsp
<context root>\private\common\sidebar.jsp
<context root>\private\jobs\selectjobscontent.jsp
<context root>\private\newrequest\newrequestpage.jsp
<context root>\private\options\optionspage.jsp
<context root>\private\query\createpage.jsp
<context root>\private\query\runpage.jsp

Example If subpage is defined in a tabpanel selectedTabParameter attribute, the following tag:

```
<actabpanel:tab key="_scheduled">
```

provides the ability to select this tab by using the following URI:

```
http://<application server>:<port>/iportal/selectjobs.do  
?subpage=_scheduled
```

tabBegin

Specifies HTML or JSP code to execute before defining the first tab in a tab panel.

Library actabpanel

Tag class com.actuate.activeportal.tags.tabpanel.TabBegin

Attributes There are no attributes for this tag. Place the desired code as the body of the tag.

Used in <context root>\private\common\errors\error.jsp
<context root>\private\common\sidebar.jsp
<context root>\private\newrequest\newrequestpage.jsp
<context root>\private\options\optionspage.jsp

```
<context root>\private\query\createpage.jsp
<context root>\private\query\runpage.jsp
```

Example The following example specifies the inclusion of several images to create a border with rounded edges before defining the tabs:

```
<ui:tabBegin>
  <TR>
    <TD>"
      width=8>
    </TD>
    <TD>"
      width=100%>
    </TD>
    <TD>"
      width=8>
    </TD>
  </TR>
</ui:tabBegin>
```

tabEnd

Specifies HTML or JSP code to execute after defining the last tab in a tab panel.

Library actabpanel

Tag class com.actuate.activeportal.tags.tabpanel.TabEnd

Attributes There are no attributes for this tag. Place the desired code as the body of the tag.

Used in <context root>\private\common\errors\error.jsp
 <context root>\private\common\sidebar.jsp
 <context root>\private\newrequest\newrequestpage.jsp

Example The following example specifies the inclusion of several images to create a border with rounded edges after defining the tabs:

```
<ui:tabEnd>
  <TR>
    <TD>"
      width=8>
    </TD>
    <TD>"
      width=100%>
    </TD>
  </TR>
```

tabMiddle

```
</TD>
<TD><img border=0 height=8
      src=<html:rewrite page="/images/bottom_r_corner.gif"/>
      width=8>
</TD>
</TR>
</ui:tabEnd>
```

tabMiddle

Specifies HTML or JSP code to execute for each currently unselected tab.

Library	actabpanel
Tag class	com.actuate.activeportal.tags.tabpanel.TabMiddle
Attributes	There are no attributes for this tag. Place the desired code as the body of the tag.
Used in	<pre><context root>\private\common\errors\error.jsp <context root>\private\common\sidebar.jsp <context root>\private\jobs\selectjobscontent.jsp <context root>\private\newrequest\newrequestpage.jsp <context root>\private\options\optionspage.jsp <context root>\private\query\createpage.jsp <context root>\private\query\runpage.jsp</pre>
Example	The following example specifies the color, width, alignment, and other attributes of unselected tabs:

```
<ui:tabMiddle>
  <TD bgcolor="#31659C" width=7>&nbsp;&nbsp;&nbsp;</TD>
  <TD bgcolor="#31659C" class="cellSidebar" valign="center"
    nowrap="nowrap">
    <A href="<%= request.getContextPath() %>/{2}"
      class="lnkSidebar">{0}</A>
  </TD>
  <TD bgcolor="#31659C" width=7>&nbsp;&nbsp;&nbsp;</TD>
</ui:tabMiddle>
```

tabMiddleSelected

Specifies HTML or JSP code to execute for the currently selected tab.

Library	actabpanel
Tag class	com.actuate.activeportal.tags.tabpanel.TabMiddleSelected
Attributes	There are no attributes for this tag. Place the desired code as the body of the tag.

Used in <context root>\private\common\errors\error.jsp
 <context root>\private\common\sidebar.jsp
 <context root>\private\jobs\selectjobscontent.jsp
 <context root>\private\newrequest\newrequestpage.jsp
 <context root>\private\options\optionspage.jsp
 <context root>\private\query\createpage.jsp
 <context root>\private\query\runpage.jsp

Example The following example specifies the color, width, alignment, and other attributes of the selected tab:

```
<ui:tabMiddleSelected>
  <TD bgcolor="#31659C" width=7>&nbsp;&nbsp;&nbsp;</TD>
  <TD bgcolor="#31659C" class="cellSidebarSelected"
    nowrap="nowrap">
    <A href="<%= request.getContextPath() %>/{2}"
      class="lnkSidebarSelected">{0}</A>
  </TD>
  <TD bgcolor="#31659C" width=7>&nbsp;&nbsp;&nbsp;</TD>
</ui:tabMiddleSelected>
```

tabPanel

Defines a tab panel and the pages associated with each tab. The tabPanel tag contains other tags from the actabpanel library that specify different parts of the tab panel.

Library actabpanel

Tag class com.actuate.activeportal.tags.tabpanel.TabPanelTag

Attributes Table 7-27 lists and describes the attributes for tabPanel.

Table 7-27 Attributes for tabPanel

Attribute	Required	Description
content Attribute	No	Specifies any HTML attributes to apply to the page part of the HTML table if style=vertical.
defaultTab	No	The key of the tab to select if selectedTab is null. If selectedTab and defaultTab are unspecified, the first tab becomes the selected tab.
flush	No	Specifies whether the server should start writing the server response before processing the entire page.

(continues)

Table 7-27 Attributes for tabPanel (continued)

Attribute	Required	Description
selectedTab	No	Specifies the key of the desired tab. This causes highlighting of the selected tab and display of the page associated with the tab.
selectedTab Parameter	No	Specifies the parameter name that URIs use to specify the key of the desired tab.
style	No	Specifies whether the tab panel is horizontal or vertical.
tabAttribute	No	Specifies any HTML attributes to apply to the tab part of the HTML table if style=vertical.
tableAttribute	No	Specifies any HTML attributes to apply to the nested HTML table containing the tabs.

Used in <context root>\private\common\errors\error.jsp
 <context root>\private\common\sidebar.jsp
 <context root>\private\jobs\selectjobscontent.jsp
 <context root>\private\newrequest\newrequestpage.jsp
 <context root>\private\options\optionspage.jsp
 <context root>\private\query\createpage.jsp
 <context root>\private\query\runpage.jsp

Example The following example creates a tab panel with four tabs. The _completed tab is chosen by default and URLs can specify the tab desired by using subpage=<tab key>:

```
<ui:tabPanel
  selectedTabParameter="subpage" defaultTab="_completed" >
  <ui:tab key="_scheduled">
    <bean:message key="TAB_SCHEDULES"/>
    <ui:content page="scheduledjob.jsp"/>
  </ui:tab>
  <ui:tab key="_pending" >
    <bean:message key="TAB_PENDING"/>
    <ui:content page="pendingjob.jsp"/>
  </ui:tab>
  <ui:tab key="_running" >
    <bean:message key="TAB_RUNNING"/>
    <ui:content page="runningjob.jsp"/>
  </ui:tab>
```



```

<ui:tab key="_completed" >
    <bean:message key="TAB_COMPLETED"/>
    <ui:content page="completedjob.jsp"/>
</ui:tab>
</ui:tabpanel>

```

tabSeparator

Specifies HTML or JSP code to execute between defining each adjacent pair of tabs.

Library	actabpanel
Tag class	com.actuate.activeportal.tags.tabpanel.TabSeparator
Attributes	There are no attributes for this tag. Place the desired code as the body of the tag.
Used in	<pre> <context root>\private\common\errors\error.jsp <context root>\private\common\sidebar.jsp </pre>
Example	The following example specifies the inclusion of several images to create a dividing line between the tabs:

```

<ui:tabSeparator>
    <TR style="width: 100%">
        <TD colspan=3>
            
                "width=100% height=8 border=0>
        </TD>
    </TR>
    <TR style="width: 100%">
        <TD colspan=3>
            "
                width=100% height=1 border=0>
        </TD>
    </TR>
    <TR style="width: 100%">
        <TD colspan=3>
            
                width=100% height=8 border=0>
        </TD>
    </TR>
</ui:tabSeparator>

```

tabSeparator

Actuate Information Console JavaBeans

This chapter contains the following topics:

- Information Console JavaBeans overview
- Information Console JavaBeans package reference
- Information Console JavaBeans class reference
- Information Console UserInfoBean class reference

Information Console JavaBeans overview

This section describes the Information Console JavaBeans. Information Console JavaBeans provide functionality, business logic, and dynamic content to Information Console web applications. Information Console JavaBeans are in aciportal.jar, which resides in <context root>\WEB-INF\lib.

The Javadoc is provided for the JavaBeans in <Actuate product root>\iServer\servletcontainer\mgmtconsole\help\api. Refer to the Javadoc for a list of JavaBean methods and their arguments.

Information Console JavaBeans package reference

Table 8-1 lists and describes the Actuate packages used in Information Console.

Table 8-1 Information Console packages

Package	Contents
com.actuate .activeportal.beans	JavaBeans that maintain information used by the Action classes.
com.actuate .activeportal.forms	JavaBeans derived from the Jakarta Struts org.apache.struts.action.ActionForm object. These JavaBeans store and validate the request parameters in HTTP requests.
com.actuate .activeportal.list	An interface, IContentList, that defines the behavior of lists of items such as files and channels. Several classes in com.actuate.activeportal.forms use this interface.

Information Console JavaBeans class reference

This section lists and describes the Information Console JavaBean classes by topic.

Channels

Table 8-2 lists and describes Information Console com.actuate.activeportal.forms classes that support channels.

Table 8-2 Channel classes

Class	Description
ChannelListAction Form	Provides the list of channels to which the user subscribes or has available.

Table 8-2 Channel classes

Class	Description
GeneralFilter ActionForm	Serves as a base ActionForm for several other ActionForms. Provides methods that handle filters to select which items the Actuate BIRT iServer returns. For example, you can request all folders and only the most recent version of all executable files.
SubscribeChannel ActionForm	Stores a list of channels available to the user, including unsubscribed channels.

Cubes, information objects, and queries

Table 8-3 lists and describes Information Console `com.actuate.activeportal.beans` classes that support cubes, information objects, and queries.

Table 8-3 Cube, information object, and query classes

Class	Description
CreateQueryBean	Contains Actuate Query information used to create, edit, or run a query (.dov) from a data source (.dox). <code>AcCreateQueryAction</code> and <code>AcRunQueryAction</code> store this JavaBean as a session JavaBean with <code>createQueryBean</code> as the attribute name. Those action classes use the IDAPI <code>getQuery</code> method to get a query definition from a DOX. They then set the query definition for this JavaBean using <code>CreateQueryBean</code> 's <code>setQueryDefinition()</code> .
CubeParam	Used by <code>AcViewCubeAction</code> class to store the names and values of parameters in cubes.
JobActionForm	Serves as a base ActionForm for <code>QueryActionForm</code> and <code>SubmitJobActionForm</code> . Stores information about the document, parameters, schedule, and other options in submitting a job or query.
SummaryBean	Stores a summary item displayed in the summary tab in Actuate Query. Summary items use functions such as sum average, minimum, and so on.

Table 8-4 describes the Information Console `com.actuate.activeportal.forms` class that supports queries.

Table 8-4 Query form class

Class	Description
QueryActionForm	Stores information to create, edit, run, or submit an Actuate query. The query action class processes the form and stores the information to <code>createQueryBean</code> , a session JavaBean.

Documents

Table 8-5 lists and describes Information Console `com.actuate.activeportal.forms` classes that support the Document pages.

Table 8-5 Document classes

Class	Description
BrowseFileActionForm	Supports browsing through the available files, including using filters to search.
CreateFolderActionForm	Supports creating a folder in the Encyclopedia volume.
FileFoldersPrivilegeActionForm	Stores information about file and folder access rights, the available users and roles, and so forth. Information Console uses this information to set up file and folder privileges.
FileListActionForm	Retrieves a list of folders or files. This ActionForm supports setting filters specifying characteristics of objects. Stores the most recent list of items returned from iServer.
GeneralFilterActionForm	The base ActionForm for several other ActionForms. Provides methods that handle filters to select which items the iServer returns. For example, you can request all folders and only the most recent version of all executable files.
GetFileDetailsActionForm	Stores the details of a file or folder. <code>AcGetFileDetailsAction</code> gets the details and stores them in this JavaBean.
SearchFilesActionForm	Stores information about the filter set by the user in the Search page. Jakarta Struts uses the filter to retrieve the list of files from the iServer and store them in this form.

General

Table 8-6 describes the Information Console `com.actuate.activeportal.beans` class that supports general functionality.

Table 8-6 General bean class

Class	Description
LinkBean	Generates an HTML link tag using the <code>link</code> , <code>linkAttributes</code> , and <code>text</code> properties. By default, the link class is <code>hyperlink</code> . After setting these properties, use the <code>toString()</code> method to generate an HTML link tag in the following format: <code>text</code>

Table 8-7 lists and describes Information Console `com.actuate.activeportal.forms` classes that support general functionality.

Table 8-7 General forms classes

Class	Description
BaseActionForm	The base ActionForm for all other Information Console ActionForms. Provides methods related to postback.
PingActionForm	Stores information used by the ping action. Ping detects the status of Information Console and iServer communication.

Jobs

Table 8-8 lists and describes Information Console com.actuate.activeportal.forms classes that support jobs.

Table 8-8 Job classes

Class	Description
GeneralFilterActionForm	Serves as a base ActionForm for several other ActionForms. Provides methods that handle filters to select which items the iServer returns. For example, you can request all folders and only the most recent version of all executable files.
GetJobDetailsActionForm	Stores detail information on jobs. AcGetJobDetailsAction uses this class to store and retrieve the job detail information for display.
JobActionForm	The base ActionForm for QueryActionForm and SubmitJobActionForm. Stores values used in submitting a job or query, such as the document, parameters, and schedule.
SelectJobNoticesActionForm	Stores the list of job notices for a channel.
SelectJobsActionForm	Contains the list of job properties for a scheduled, running, pending, or completed job.
SubmitJobActionForm	Contains the information for submitting a job from the requester page. This class extends JobActionForm.

Skins

Table 8-9 lists and describes Information Console com.actuate.activeportal.beans classes that support skins.

Table 8-9 Skin bean classes

Class	Description
GroupBean	Stores lists of all images, colors, fonts, and styles for a skin. Each list is a list of SkinBean objects.

(continues)

Table 8-9 Skin bean classes (continued)

Class	Description
SkinBean	Stores information about an image, style, color, or font. The information for this JavaBean comes from a <Style> or <Image> tag in the skin.config file for a skin. Access this information using the <code>getStyle()</code> or <code>getImage()</code> methods. SkinBeans are grouped into GroupBeans for each skin.
SkinManagerInfoBean	Stores access information about a skin. Used by the <code>SkinManagerActionForm</code> .

Table 8-10 lists and describes Information Console `com.actuate.activeportal.forms` classes that support skins.

Table 8-10 Skin form classes

Class	Description
FileUploadActionForm	Uploads images during skin customization and stores an object representation of the uploaded file. It uses Jakarta Struts <code>org.apache.struts.upload.FormFile</code> to handle file upload. The file is saved in a temporary folder on the server.
SkinEditorActionForm	Stores all the information about the various groups defined in the skin.config file. When an administrator edits a skin, Information Console loads the skin.config file and represents its contents as a <code>SkinConfig</code> object. Changes to the skin's images, color, and fonts are stored in GroupBeans for a skin.
SkinManagerActionForm	Stores the list of available skins. Use the <code>getSkin()</code> method to get the list of skins as a <code>Vector</code> of <code>SkinManagerInfoBean</code> . This form supports adding, cloning, and deleting skins.

Users

Table 8-11 lists and describes Information Console `com.actuate.activeportal.beans` classes that support handling users.

Table 8-11 User bean classes

Class	Description
FeatureOptionsBean	Stores the features available to the current user. It contains Information Console functionality levels and reporting features on the iServer the user is using. Access this class using <code>UserInfoBean.getFeatureBean()</code> .
ProfileBean	Stores the user profile settings obtained from the iServer. Access this class by using <code>UserInfoBean.getProfile()</code> .

Table 8-11 User bean classes

Class	Description
UserAgentBean	Detects what kind of browser the user is using from the HTTP header user-agent. After instantiating this JavaBean, you must call setRequest(HttpServletRequest request). Get the browser type by calling isIE(), isNS4(), and isNS6() methods.
UserInfoBean	Contains information about the user, such as the user's Encyclopedia volume name, iServer URL, preferred skin, and authentication ID assigned by the iServer. Several methods also affect the display and highlighting of features.

Table 8-12 lists and describes Information Console com.actuate.activeportal.forms classes that support handling users.

Table 8-12 User form classes

Class	Description
LoginForm	Stores information about the user ID, server URL, volume, and other information specified during login.
UserOptionsActionForm	Stores the selected choices on the options page, including the skin, view, experience level, and e-mail ID. This form supports changing these options.

Information Console UserInfoBean class reference

Table 8-13 lists and describes the methods other than set methods available in the Information Console com.actuate.activeportal.beans.UserInfoBean class.

Table 8-13 UserInfoBean methods

Method	Description
getAcLocale()	Gets the AcLocale object specifying the Actuate locale for the current user.
getAdminRights()	Gets the administrator rights of the current user. If the user is not an administrator or operator, this method returns null. An administrator or application sets these rights when creating a user.
getAuthid()	Gets a String containing the authentication ID returned by the iServer for this user during login. Use this authentication ID in IDAPI calls.

(continues)

Table 8-13 UserInfoBean methods (continued)

Method	Description
getCurrentfolder()	Gets a string containing the name of the most recent folder accessed by the user.
getDefaultAnalyticsExpLevel()	Gets a string for the default Actuate Analytics experience level for this user. The level is Novice unless overridden on the options page or by calling setDefaultAnalyticsExpLevel().
getDefaultServerURL()	Gets a string for the URL to use for the default server for the user.
getDefaultVolume()	Gets the volume name from the VOLUME_DEFAULT tag in <context root>\WEB-INF\web.xml.
getFeatureOptionsBean()	Gets a JavaBean that stores the features and iServer options that are available to the current user.
getFeatures()	Gets a list of all features defined in the functionality-level.config file.
getFilter()	Gets a string containing the filter the user most recently typed into the search field of the Documents page. If the user has not typed a filter, this method returns null.
getHomefolder()	Gets a string specifying the user's home folder. An administrator or application sets this value when creating a user.
getIportalid()	Gets a string specifying the Information Console session id.
getLocale()	Gets the current login user's java.util.Locale object.
getMaxJobPriority()	Gets the maximum job priority permitted for this user. An administrator or application sets this value when creating a user.
getOnlylatest()	Gets the string "true" if the filter on the Documents page specifies showing only the most recent version of each file.
getPassword()	Gets a string containing the user's password.
getProfile()	Gets the ProfileBean. This JavaBean stores information about the user's settings on the Information Console options page. This information includes current skin, view, experience level, and so on.
getProperty(java.lang.String name)	Gets a string containing the value of a custom property having the name passed as a parameter. Create custom properties and set their values using setProperty().
getRepositoryType()	Gets a string specifying the type of repository that the user is accessing as: workgroup: local file system enterprise: an Encyclopedia volume

Table 8-13 UserInfoBean methods (continued)

Method	Description
getRoleNames()	Gets an array of strings containing a list of the user's feature roles, such as Actuate Information Console Intermediate, Actuate Information Console Advanced, or Actuate Information Console Administrator.
getServerurl()	Gets the URL of the server to which the current user is logged in. This URL includes the protocol and the port. For example: http://localhost:9000.
getShowdocuments()	Gets the string "true" if the filter on the Documents page specifies including documents.
getShowexecutables()	Gets the string "true" if the filter on the Documents page specifies including executable files.
getShowfolders()	Gets the string "true" if the filter on the Documents page specifies including folders.
getSideBarFeatures()	Gets the list of features available to this user on the side menu, tabs, the tree, or equivalent structure. Some features, such as customization, are not part of this set.
getSidebarSelected()	Gets the URL for the feature selected on the side menu, tab, tree, or equivalent structure. This method is used to highlight a feature in the sidebar.
getSkinConfig()	Gets the SkinConfig object for the user's current skin. The SkinConfig object contains all information defined for the skin.
getSkinName()	Gets a string containing the name of the skin used by the user.
getSubfeatures()	Gets a collection containing a list of all subFeatures defined in <context root>\WEB-INF\functionality-level.config.
getSystemname()	Gets a string containing the name of the iServer machine.
getTimezone()	Gets the AcTimeZone object specifying the time zone for the current user.
getUserAgent()	Gets the UserAgentBean object for the user. UserAgentBean detects the user's browser type.
getUserid()	Gets the userID of the current user.
getView()	Gets the string specifying the current view for this user.
getVolume()	Gets the string specifying the Encyclopedia volume that the user is accessing.
init()	Initializes the UserInfoBean members.

(continues)

Table 8-13 UserInfoBean methods (continued)

Method	Description
isAlwaysGetFolderList()	Returns True if the Documents page should always show the folder list, even if it is not selected on the filter.
isEanalysisOptionEnabled()	Returns True if the Actuate e.Analysis option is enabled for the user.
isHomeFolderSet()	Returns True if the user has a home folder specified in the Encyclopedia volume.
isShowFilters()	Returns True if the filter panel is shown for all lists of documents, jobs, and channels.
isViewInNewBrowser Window()	Returns True if the report viewer is specified to launch in a new browser window.
toString()	Returns a string representation of the object.

UserInfoBean calls set methods when the user logs in to set the values that the get methods return. Typically, your application should not call the set methods as the bean would then be inconsistent with the information stored in the repository or external security application. These set methods only change the values in the bean, so the results of the calls are not deterministic.

Table 8-14 lists and describes set methods that are available in the Information Console `com.actuate.activeportal.beans.UserInfoBean` class.

Table 8-14 UserInfoBean set methods

Method	Description
setAcLocale(<code>com.actuate.reportcast.utils.AcLocale acLocale</code>)	Sets the Actuate locale for the current user with the specified <code>AcLocale</code> object. Also changes the Java locale.
setAlwaysGetFolderList (<code>boolean b</code>)	Set to True if the Documents page should always show the folder list, even if it is not selected on the filter.
setAuthid(<code>java.lang.String authid</code>)	Sets the authentication ID to the string passed in as a parameter. The authentication ID is returned by the Actuate BIRT iServer and set for the user during login. Use <code>getAuthid()</code> to use this authentication ID in IDAPI calls.
setCurrentfolder(<code>java.lang.String currentfolder</code>)	Sets the string specifying the most recent folder name accessed by the user.
setDefaultAnalyticsExp Level(<code>java.lang.String analyticsExpLevel</code>)	Sets the default Actuate Analytics experience level for this user. The level is Novice unless overridden on the options page or by calling this method.

Table 8-14 UserInfoBean set methods (continued)

Method	Description
setDefaultServerURL(java.lang.String defaultServerURL)	Sets the URL to use as a default value for users.
setDefaultVolume(java.lang.String defaultVolume)	Sets the volume to use if no volume name is specified by the URL in the request. By default, Information Console sets the default volume to the value in the VOLUME_DEFAULT tag in <context root>\WEB-INF\web.xml.
setFeatureOptions (FeatureOptionsBean featureOptionsBean)	Sets a list of all Information Console features and iServer options that are available to the current user.
setFilter(java.lang.String filter)	Sets the string specifying the filter to use as a default value in the Documents page. Information Console sets this String to the filter that the user most recently typed into the search field of the Documents page.
setHomefolder(java.lang.String string)	Sets the string specifying the user's home folder. An administrator or application sets this value when creating a user.
setMaxJobPriority(int priority)	Sets the maximum job priority permitted for this user. An administrator sets this value for a user.
setOnlylatest(java.lang.String onlylatest)	Sets value to indicate if only latest version of the documents are to be displayed in the file folder list. "true" sets Information Console to show only the most recent version of each file.
setPassword(java.lang.String password)	Sets the password to the value of the string passed as a parameter.
setProfile()	Sets the ProfileBean. This JavaBean stores information about the user's settings on the Information Console options page. This information includes current skin, view, experience level, and so on.
setProperty(java.lang.String name, java.lang.String value)	Sets the value of a custom property. Create custom properties and set their values using this method. The parameters are the name of the custom property and the value to set.
setRoleNames(java.lang.String[] strings[])	Sets a list of the user's feature roles, such as Active Portal Intermediate, Active Portal Advanced, or Active Portal Administrator.
setServerurl(java.lang.String surl)	Sets the server URL currently used by the user. This URL includes the protocol and the port, for example: http://localhost:9000.

(continues)

Table 8-14 UserInfoBean set methods (continued)

Method	Description
setShowdocuments(java.lang.String showdocuments)	Sets the value to indicate if documents are to be displayed in the file folder list. "true" sets Information Console to display documents.
setShowexecutables(java.lang.String showexecutables)	Sets the value to indicate if executables are to be displayed in the file folder list. "true" sets Information Console to display executables.
setShowFilters(boolean showFilters)	Set to True to specify that Information Console display the filter panel for all pages showing lists of files, jobs, or channels.
setShowfolders(java.lang.String showfolders)	Sets the value to indicate if folders are to be displayed in the file folder list. "true" sets Information Console to display folders.
setSideBarFeatures(com.actuate.activeportal.functionality.config.Feature[] feature)	Sets the list of features available to this user on the side menu, tabs, the tree, or equivalent structure. This list is a subset of the features available to the user.
setSidebarSelected(java.lang.String sidebarSelected)	Sets the feature highlighted on the side menu, tab, tree, or equivalent structure. To highlight a feature, pass a string containing the URI invoked by the feature. To not highlight any features, pass a string, such as "No highlighting", that does not match the URI for any feature in the side menu. By default, Information Console highlights the Documents feature.
setSkinConfig(com.actuate.activeportal.skin.SkinConfig config)	Sets the SkinConfig object for the user's current skin. The SkinConfig object contains all information defined for the skin.
setSkinName(java.lang.String string)	Sets the name of the skin used by the user.
setSystemname(java.lang.String systemName)	Sets the iServer system name to the value of the string parameter.
setTimezone(com.actuate.reportcast.utils.AcTimeZone timezone)	Sets the AcTimeZone object specifying the user's time zone.
setUserAgent(UserAgentBean userAgent)	Sets the UserAgentBean for this user. The UserAgentBean specifies the user's browser type.
setUserid(java.lang.String userid)	Sets the user ID for the user.

Table 8-14 UserInfoBean set methods (continued)

Method	Description
setView(java.lang.String string)	Sets the current view for the user. The string contains the name of the constant for the desired view. The available constants are: <ul style="list-style-type: none">■ AcConstants.VIEW_CATEGORY■ AcConstants.VIEW_LIST■ AcConstants.VIEW_DETAIL■ AcConstants.VIEW_ICON
setViewInNewBrowserWindow(boolean _newWindow)	Set to True to specify that the report viewer launch in a new browser window.
setVolume(java.lang.String volume)	Sets the value of the string specifying the name of the Encyclopedia volume the user is accessing.

Using Actuate Information Console security

This chapter contains the following topics:

- About Actuate Information Console security
- Protecting corporate data
- Understanding the authentication process
- Creating a custom security adapter
- Creating an upload security adapter

About Actuate Information Console security

A reporting web application is accessible to any user who has a web browser and the URI for the application. This chapter discusses the Actuate Information Console security features and how to use them to:

- Ensure that users access only those objects in the Encyclopedia volume for which they have permission.
- Protect sensitive reports.

The types of security you can provide for Information Console are:

- Default user authentication. Use the default Information Console and Actuate BIRT iServer facilities to ensure that users access only those reports and other Encyclopedia volume items for which they have permission.
- User authentication using the Information Console Security Extension (IPSE). Use IPSE to customize and control the user login and authentication process. For details about implementing custom user authentication, see “Creating a custom security adapter,” later in this chapter.

Protecting corporate data

iServer provides a structured content generation solution for web applications. Deploying Actuate applications developed for the internet, such as Information Console, requires planning for network security.

Internet applications support access to information within an organization from outside that organization. Because the organization’s internal network is connected to the internet, there is the risk of unauthorized access to the corporate network and to the data that resides on that network.

Organizations use one or a combination of the technologies described in the following sections to prevent unauthorized access to the corporate network and protect authentication transactions from intrusion.

Protecting corporate data using firewalls

Typically companies use firewalls to prevent unauthorized access to corporate networks and data. A firewall is a system or group of systems that restrict access between two networks, such as an organization’s internal network and the internet. Firewalls keep unauthorized users out. As a result, firewalls prevent damage caused by malicious programs such as worms and viruses from spreading to other parts of your network. At the same time, firewalls allow legitimate business to tunnel through the firewall and be efficiently conducted on your network.

Firewalls can be used to restrict access between two internal networks, for example, the accounting and engineering networks. Security teams configure firewalls to allow traffic using specific protocols, such as HTTP, over specific network addresses and ports. Be sure that your firewall allows access for the Information Console and iServer ports. For more information about the Actuate ports, see *Configuring BIRT iServer*.

Protecting corporate data using Network Address Translation

Companies also use Network Address Translation (NAT). NAT routers and software support private networks using unregistered, private IP (Internet Protocol) addresses to connect to the internet.

Protecting corporate data using proxy servers

Proxy servers, specialized web servers or hardware that operate on or behind a firewall, improve efficient use of network bandwidth and offer enhanced network security. For more information about proxy servers and Information Console, see Chapter 1, “Introducing Actuate Information Console.”

Understanding the authentication process

The authentication process involves the following steps, in this order:

- A user or client makes a request by choosing a link on an Information Console page or by typing an Actuate Information Console URI in a web browser. The Information Console application processes the request.
- Information Console checks the URI for the `forceLogin` parameter. If the `forceLogin` parameter is set to “true” in the URI, the application activates the Information Console Login page, even if the user has already logged in. If `forceLogin` is set to “false” or does not appear, the request process continues. For details about the `forceLogin` URI parameter, see “Common URI parameters” in Chapter 4, “Actuate Information Console URIs.”
- Information Console authenticates the user for the Encyclopedia volume. If the login information is invalid, the login screen appears in the browser.

If a custom security adapter parameter is set in the `web.xml` file, Information Console attempts to load the custom security adapter class. If the class loads successfully, the following steps occur:

- Information Console calls the custom security adapter’s `authenticate()` method with the parameters that the browser sent.
- The `authenticate()` method performs the custom validation.

- Information Console calls the required `getUserName()`, `getPassword()`, and `getVolumeProfile()` methods to retrieve the user information needed by the iServer.
- Optionally, Information Console calls the `getExtendedCredentials()` method. If this method returns null, there are no extended credentials to send to the iServer.
- Information Console now has all the information that it requires for connecting to the iServer. Information Console creates the necessary SOAP message for connecting to the iServer and sends a login request.

Information Console uses the default Volume Profile setting if the server, volume, or volume profile

Creating a custom security adapter

The Information Console security adapter enables other applications to authenticate users and log in to the Information Console application, for example, by using a URL. A custom security adapter can define alternate authentication requirements. In this way, an Information Console security adapter establishes an additional layer of logic to the existing Information Console authentication, as shown in Figure 9-1.

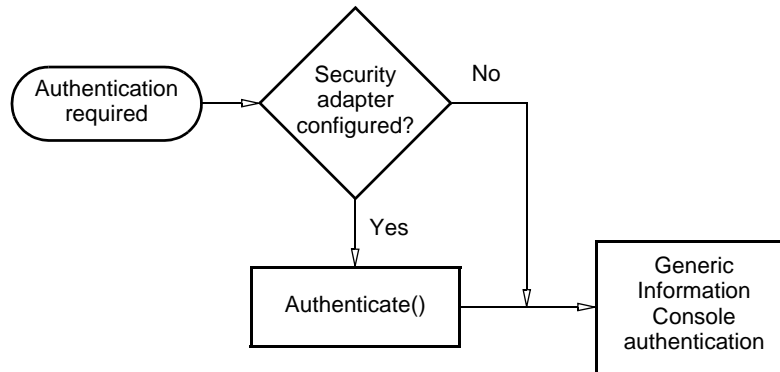


Figure 9-1 Information Console authentication system

A user cannot update their password from Information Console if a custom security adapter class is set. In this way, Information Console prevents conflicts between the user's current password and the security system that is used to verify passwords.

To create a custom security adapter, perform the following steps:

- Ensure that your application can access the IPSE Java classes.

- Create a java class that implements the custom security adapter class for IPSE.
- Deploy the Custom Security Adapter to Information Console.

Accessing the IPSE Java classes

The Information Console library, `com.actuate.iportal.jar`, contains the IPSE Java classes. This library is located in the `lib` subdirectory in the Information Console installation. The class, `com.actuate.iportal.security.iPortalSecurityAdapter`, in this library provides the framework for custom authentication. A custom security adapter providing an IPSE implementation extends this class.

Specifically, the JRE needs to access the following jars:

- `<context root>\WEB-INF\lib\com.actuate.iportal.jar`
- `<context root>\WEB-INF\lib\org.apache.xerces_<version>.jar`
- `<context root>\WEB-INF\lib\com.actuate.webcommon.jar`
- `<iPortal installation directory>\lib\servlet-api.jar`
- `<iPortal installation directory>\lib\jsp-api.jar`

Creating a custom security adapter class

Extend the `iPortalSecurityAdapter` class to customize authentication. The `iPortalSecurityExtension` requires access to the following libraries:

- `javax.servlet.http.*`
- `com.actuate.iportal.security.iPortalSecurityAdapter`

`iPortalSecurityAdapter` provides a set of empty methods. Extend this class and override any of the methods to provide custom IPSE authentication. To establish a secure session with Information Console using a custom security adapter, the following methods are required:

- A constructor
- `authenticate()`
- `getPassword()`
- `getUserName()`

The login module of Information Console calls methods in the custom security class to perform authentication and to retrieve login credentials to pass to `iServer`. The `authenticate()` method returns a boolean value to indicate whether the login credentials provided are acceptable. The getter methods return the credentials that `iServer` requires. Each user name and password must correspond to an authentic user account on the volume configured by the volume profile. If a volume profile isn't set by the security adapter, authentication uses the default

volume profile configuration. For example, to support a URL that authenticates using a single parameter, code, override `authenticate()` to retrieve the parameter from the `HttpServletRequest` and set the user name, password, and volumeProfile as in the following class:

```
import javax.servlet.http.*;
import com.actuate.iportal.security.iPortalSecurityAdapter;

public class SecurityCode extends
    com.actuate.iportal.security.iPortalSecurityAdapter {
    private String volumeProfile = "CustomAccess";
    private String userName = null;
    private String password = null;
    public SecurityCode() {}

    public boolean authenticate(
        HttpServletRequest httpServletRequest) {
        String param = httpServletRequest.getParameter("code");
        boolean secured = true;
        if ("12345".equalsIgnoreCase( param )) {
            userName = "user1";
            password = "user1";
        }
        else if ("abc".equalsIgnoreCase( param )) {
            userName = "BasicUser";
            password = "";
        }
        else {
            secured = false;
        }
        return secured;
    }
    public String getUserName() { return userName; }
    public String getPassword() { return password; }
    public String getVolumeProfile() { return volumeProfile; }
}
```

If there is a user "user1" with the password "user1" on the volume configured by the volume profile "CustomAccess," a valid URL that authenticates user1 using this security adapter is as follows:

`http://localhost:8700/iportal/getfolderitems.do?code=12345`

Deploying a custom security adapter

To deploy a custom security adapter, the Information Console application must have access to the class compressed into a JAR file. To meet this requirement, compile the class, compress it into a JAR, and move it into the `<context root>\WEB-INF\lib` directory for your Information Console application. Then, add the

class's name as the value for the SECURITY_ADAPTER_CLASS parameter in <context root>\WEB-INF\web.xml. Finally, restart the application service running Information Console to activate this change.

How to deploy a custom security adapter to Information Console

- 1 Compile the IPSE application. Use a command similar to this one in a console window:

```
javac SecurityCode.java
```

- 2 Create a JAR file to contain the IPSE application. Use a command similar to this one in a console window:

```
jar cvf SecurityCode.jar SecurityCode.class
```

- 3 Using Windows Explorer, copy SecurityCode.jar to this directory:

```
<your application context root>\WEB-INF\lib
```

- 4 Using a UTF-8 compliant code editor, open the following file:

```
<your application context root>\WEB-INF\web.xml
```

- 5 Navigate to the parameter name SECURITY_ADAPTER_CLASS.

- 6 Change the param-value parameter of the SECURITY_ADAPTER_CLASS to the fully-qualified class name for the security adapter class. Use an entry similar to this one:

```
<param-name>SECURITY_ADAPTER_CLASS</param-name>  
<param-value>SecurityCode</param-value>
```

- 7 Save and close web.xml.

- 8 Restart the application server running Information Console. For the default installation, restart the Actuate 11 Apache Tomcat for Information Console Service.

Understanding the security adapter class

To implement a custom security adapter, create a class that extends `com.actuate.iportal.security.iPortalSecurityAdapter`. This class contains the following methods.

authenticate()

Syntax	<code>boolean authenticate(javax.servlet.http.HttpServletRequest request)</code>
Parameters	request The request parameter sent from the Information Console web application.
Description	Required method that evaluates the current user's security credentials. The Login module calls <code>authenticate()</code> to validate the current user's security credentials. If <code>authenticate()</code> returns <code>False</code> , the user is redirected to the login page.

Returns True for successful credential evaluation and False otherwise.

Throws An AuthenticationException indicating the reason for the failure, if credential evaluation is not successful.

getExtendedCredentials()

Syntax byte[] getExtendedCredentials()

Description Retrieves the current user's extended security credentials.

Returns A byte array representing any extended credentials for the iServer to use to authenticate the user, or null if there are no extended credentials to evaluate.

getPassword()

Syntax String getPassword()

Description Required method that retrieves the current user's password. The Login module calls getPassword() and uses the password to establish a connection to the iServer and to access the Encyclopedia volume.

Returns A string that is the password to use to establish the connection.

getRepositoryType()

Syntax String getServerUrl()

Description Retrieves the repository type. The Login module calls this method to check the repository type. Alternatively, provide isEnterprise().

Returns A string that indicates the repository type. The repository type for iServer is enterprise.

getRunAsUser()

Syntax String getRunAsUser()

Description Retrieves the runAs setting if the runAs is enabled. The Login module calls this method to retrieve the user name used for a run as operation.

Returns A string containing the user name that corresponds to the runAs user setting.

getServerUrl()

Syntax String getServerUrl()

Description Retrieves the URL of the server to which the current user connects. The Login module calls getServerURL().

Returns A string containing the URL for the iServer currently connected.

getUserHomeFolder()

- Syntax** String getUserHomeFolder()
- Description** Retrieves the current user's home folder. The Login module calls getUserHomeFolder() to access the user's files.
- Returns** A string that is the user's home folder. It is null if there is no home folder for the user.

getUserName()

- Syntax** String getUserName()
- Description** Required method that retrieves the current user's login name. The Login module calls getUserName() to establish a connection to the iServer and to access the Encyclopedia volume.
- Returns** A string containing the user name that the iServer recognizes.

getVolume()

- Syntax** String getVolume()
- Description** Retrieves the volume to which the current user connects. The Login module calls getVolume() to retrieve the name of the Encyclopedia volume to which the user wishes to connect.
- Returns** A string containing the domain and volume name for the Encyclopedia volume to which the user connects to through the iServer. If null, the iServer connects to the default volume, read from the DEFAULT_VOLUME parameter in the Information Console web.xml file.

getVolumeProfile()

- Syntax** String getVolumeProfile()
- Description** Required method that retrieves the volume profile to which the current user connects. The Login module calls getVolumeProfile() to retrieve the name of the volume profile to which the user wishes to connect.
- Returns** A string containing the server profile name for the Encyclopedia volume to which the user connects through the iServer.

isEnterprise()

- Syntax** boolean isEnterprise()
- Description** Evaluates whether the user connects to an Encyclopedia volume. The Login module calls isEnterprise() to determine whether to use an Encyclopedia volume repository.
- Returns** True.

Creating an upload security adapter

The default security for Information Console upload functionality checks a file's type against the value of the `UPLOAD_FILE_TYPE_LIST` parameter in `web.xml`. The Information Console upload security adapter provides additional external verification features for the file upload feature using a Java interface, `com.actuate.iportal.security.IUploadSecurityAdapter`.

Information Console upload security adapter establishes an additional layer of logic to the existing Information Console authentication, as shown in Figure 9-2.

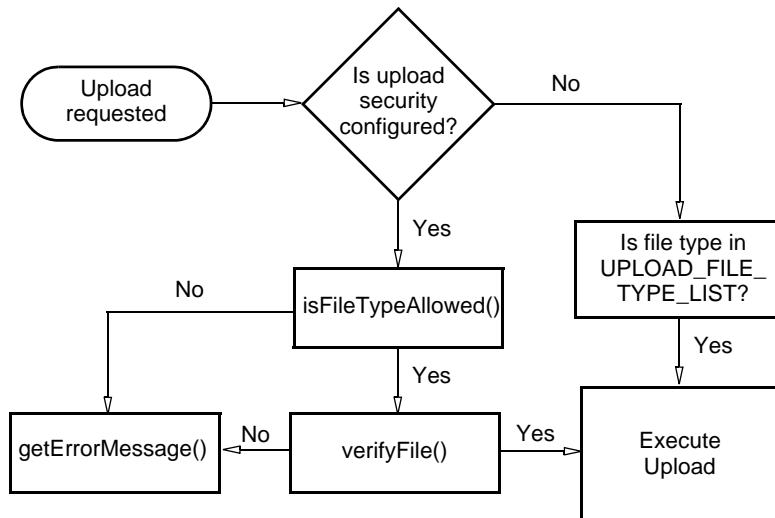


Figure 9-2 Information Console upload security system

If an upload security adapter is configured, Information Console calls `isFileTypeAllowed` to check whether the file's file type is allowed. If so, then it calls `verifyFile` to perform any additional verification steps. If either `isFileTypeAllowed` or `verifyFile` returns false, Information Console displays an error message supplied by `getErrorMessage`, or a generic message if `getErrorMessage` returns null.

To create an upload security adapter, perform the following steps:

- Ensure that your application can access the necessary Java classes.
- Create a java class that implements the upload security adapter interface.
- Deploy the upload security adapter class to Information Console.

Accessing the necessary Java classes

The Information Console library, `com.actuate.iportal.jar`, contains the security extension Java classes. This library is located in the `lib` subdirectory of the Information Console installation. The upload security adapter interface, `com.actuate.iportal.security.IUploadSecurityAdapter`, in this library provides the framework for additional upload security. A valid upload security adapter implements this interface.

Specifically, the JRE needs to access the following jars:

- `<context root>\WEB-INF\lib\com.actuate.iportal.jar`
- `<context root>\WEB-INF\lib\org.apache.xerces_2.9.0.v201005080400.jar`
- `<iPortal installation directory>\lib\servlet-api.jar`
- `<iPortal installation directory>\lib\jsp-api.jar`

Creating a custom security adapter class

Implement the upload security adapter interface to customize file verification. The upload security adapter requires access to the following libraries:

- `javax.servlet.http.HttpServletRequest`
- `javax.servlet.ServletContext`
- `com.actuate.iportal.security`

To process a secure upload request from Information Console using an upload security adapter, the following methods are required:

- `getErrorMessage()`
- `isFileTypeAllowed()`
- `verifyFile()`

For example, to prevent any file type except plain text (.txt) from being uploaded, implement `txt` as the only valid file type for `isFileTypeAllowed`, as in the following class:

```
package com.actuate.iportal.security;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;

public class SecureUpload implements IUploadSecurityAdapter {

    public boolean isFileTypeAllowed( HttpServletRequest request,
        String fileType ) {
        if ( fileType == null ) return false;
        if ( fileType.toLowerCase().trim().equals("txt") ) return true;
    }
}
```

```

        else return false;
    }

    public boolean verifyFile(HttpServletRequest request, String
        fileName, String dstFolder){
        return true;
    }

    public String getErrorMessage(HttpServletRequest request){
        String message = "Only plain text (.txt) files are permitted.";
        return message;
    }
}

```

When the upload security adapter requires file validation, Information Console copies the file temporarily into the directory specified by TEMP_FOLDER_LOCATION parameter in web.xml.

Deploying an upload security adapter

To deploy an upload security adapter, the Information Console application must have access to the class compressed into a JAR file. To meet this requirement, compile the class, compress it into a JAR, and move it into the <context root>\WEB-INF\lib directory for your Information Console application. Then, add the class's name as the value for the UPLOAD_SECURITY_ADAPTER parameter in <context root>\WEB-INF\web.xml. Finally, restart the application service running Information Console to activate this change.

How to deploy an upload security adapter to Information Console

- 1 Compile the Upload security application. Use a command similar to this one in a console window:

```
javac SecureUpload.java
```

- 2 Create a JAR file to contain the upload security application. Use a command similar to this one in a console window:

```
jar cvf SecureUpload.jar SecureUpload.class
```

- 3 Using Windows Explorer, copy SecureUpload.jar to this directory:

```
<your application context root>\WEB-INF\lib
```

- 4 Using a UTF-8 compliant code editor, open the following file:

```
<your application context root>\WEB-INF\web.xml
```

- 5 Navigate to the parameter name UPLOAD_SECURITY_ADAPTER.

- 6 Change the param-value parameter of the UPLOAD_SECURITY_ADAPTER to the fully-qualified class name for the upload security adapter class. Use an entry similar to this one:

```
<param-name>UPLOAD_SECURITY_ADAPTER</param-name>  
<param-value>SecureUpload</param-value>
```

- 7 Save and close web.xml.
- 8 Restart the application server running Information Console. For the default installation, restart the Actuate 11 Apache Tomcat for Information Console Service.

Understanding the upload security adapter interface

To implement a custom upload security adapter, create a class that implements the `com.actuate.iportal.security.IUploadSecurityAdapter` interface. This interface defines the following methods.

getErrorMessage()

- Syntax** `String getErrorMessage(javax.servlet.http.HttpServletRequest request)`
- Parameters** **request**
The request parameter sent from the Information Console web application.
- Description** A method that returns a custom error string when either `isFileTypeAllowed` or `verifyFile` returns `False`.
- Returns** `String`. An error message. If null, Information Console displays a generic default error message.

isFileTypeAllowed()

- Syntax** `boolean isFileTypeAllowed(javax.servlet.http.HttpServletRequest request, string fileType)`
- Parameters** **request**
The request parameter sent from the Information Console web application.
- fileType**
`String`. The type of the upload file, as determined by file extension.
- Description** A required method used to do additional validation of the upload file.
- Returns** `Boolean`. `True` for an allowed file type and `False` otherwise.

verifyFile()

- Syntax** `boolean verifyFile(javax.servlet.http.HttpServletRequest request, string filePath, string dstPath)`

Parameters	request The request parameter sent from the Information Console web application.
	filePath String. The path of the file stored on Information Console. The default location is the directory specified by TEMP_FOLDER_LOCATION parameter in web.xml.
	dstPath String. The repository path to which the upload sends the file.
Description	A required method used to do additional validation of the upload file.
Returns	Boolean. True for successful file validation and False otherwise.

Customizing Information Console online help

This chapter contains the following topics:

- About Actuate Information Console online help files
- Using a custom help location
- Creating a localized help collection
- Customizing icons, links, and the company logo
- Changing help content

About Actuate Information Console online help files

Actuate provides Information Console online help using the internet by default. To customize online help for Information Console, install the documentation on the local server and switch the help location for Information Console to the local server. Then, customize the online help as needed.

How to switch the help location for Information Console for Windows

Use the following procedure to switch the help location of Information Console. Switching the help location is required for any of the customization tasks detailed in this chapter.

- 1 Open the Actuate Documentation and Localization DVD and run setup.exe. Use the default settings for documentation installation to install the help for all installed Actuate products.
- 2 From the Windows Start menu, choose Programs→Actuate11→Switch Help Location.
- 3 On docupdate, select Use local help, as shown in Figure 10-1.

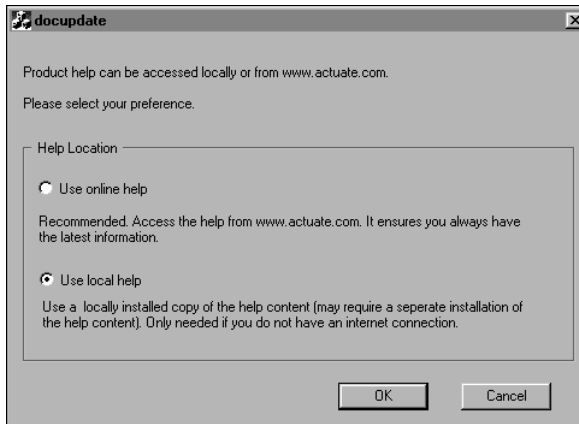


Figure 10-1 Switching the help location for Information Console
Choose OK.

- 4 Restart the service for Information Console. For a stand-alone application, this service is Actuate 11 Apache Tomcat for Information Console service.

Understanding the Information Console help directory structure

The local Information Console help files are grouped into directories under the context root for Information Console, which is the home directory in which the

Actuate product resides. For example, the default context root for Information Console installed as a component of iServer on Windows systems is <Actuate home>\iServer\servletcontainer\iportal and on UNIX and Linux systems is <Actuate home>/iServer/servletcontainer/iportal. The localized help directory under the context root is the container for the help implementation. For example, in Information Console it is:

```
<context root>\help
```

Figure 10-2 illustrates the Information Console help directory structure.

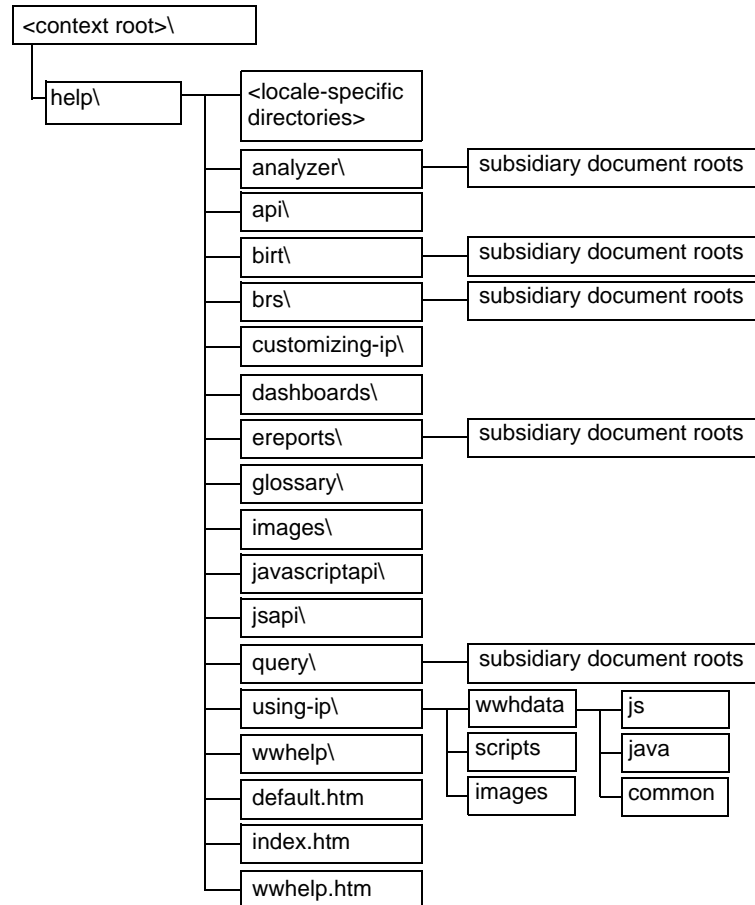


Figure 10-2 Information Console help directory structure

Actuate uses JavaScript (.js) and HTML (.html) files to implement Information Console help. The files that support top-level help styles and images reside in the wwhelp directory. Files that support help content pages and help navigation reside in a document root directory. A document root contains the help files for a specific top-level help topic, such as dashboards or glossary.

Understanding a help collection

The wwhelp directory contains files that support grouping multiple document roots into a collection. If you open the help using index.htm, the table of contents frame displays the top-level help topics, as shown in Figure 10-3.

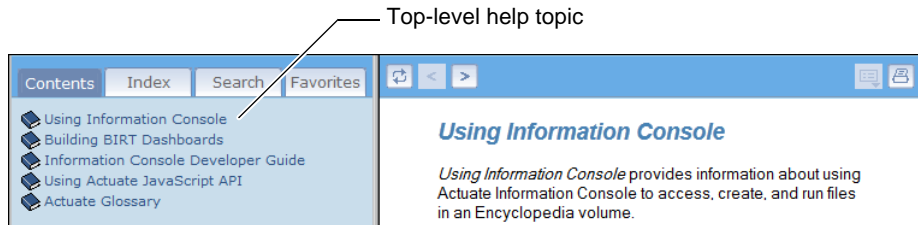


Figure 10-3 Appearance of top-level help topics

A collection has a one-to-one correlation between each top-level help topic and a document root. Each top-level help topic represents a complete book. Table 10-1 lists these applications and the directory containing the corresponding help collection.

Table 10-1 Applications and help collection directories

Application	Directory
Using Information Console	using-ip
Building and Using BIRT Dashboards	dashboards
Information Console Developer Guide	customizing-ip
Using Actuate JavaScript API	javascriptapi
Actuate Glossary	glossary

The help directory contains subdirectories that provide the help collections for applications launched by Information Console. Table 10-2 lists each document root in the Information Console Online help collection and its corresponding top-level help topic.

Table 10-2 Top-level help topics

Help topic	Document root
Actuate BIRT Viewer and Interactive Viewer	birt
Actuate Query	query
BIRT Data Analyzer	analyzer
BIRT Studio	brs
e.Reports DHTML Viewer	ereports

Understanding a document root

The content files for a top-level help topic reside in a corresponding document root. For example, the using-ip document root contains iPusing-intro.2.1.html, iPusing-intro.2.2.html, and so on. These files are the content files for the help. Each document root also contains an index.html file. Opening this file displays the topic and content files for the book.

Within each document root is a wwldata\common directory that contains the JavaScript files that organize help content and that link the help files to the application. Table 10-3 lists and describes the customizable <document root>\wwldata\common contents.

Table 10-3 Help content management files

File	Purpose
files.js	Lists the content files to be used and in what order
title.js	Specifies the title for the browser window and the top-level table of contents text
topics.js	Designates the targets for context-sensitive help keys the Information Console emits

Within each document root, a wwldata\js directory contains JavaScript files that organize the navigation frame. This frame includes the table of contents (TOC), index, and search frames. Table 10-4 lists and describes the customizable <document root>\wwldata\js contents.

Table 10-4 Help navigation files

File	Purpose
index.js	Organizes the index links and hierarchy
search.js	Designates specific search values and priority
toc.js	Specifies the table of contents frame hierarchy, linking behavior, and text

Understanding context-sensitive help

The Information Console application links to its online help files using wwhelp.html located in <context root>\help. Typically, the links that activate this context-sensitive help are in the Information Console application, as shown in Figure 10-4.

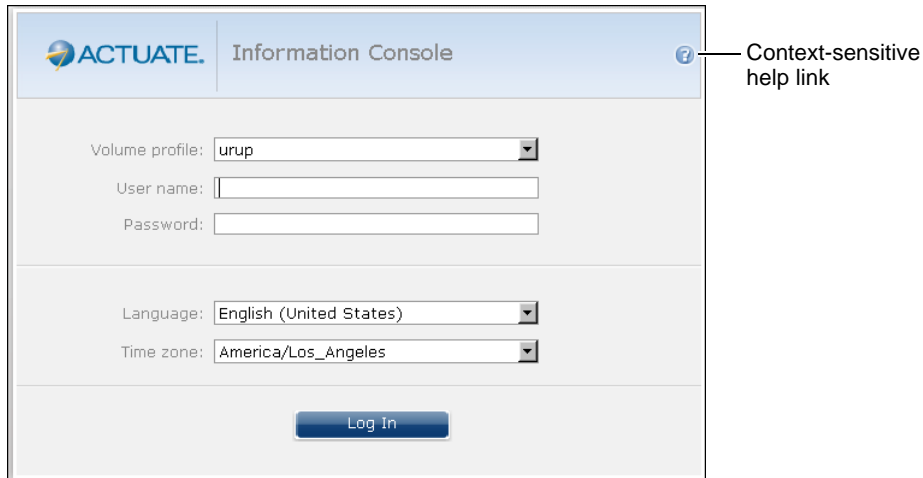


Figure 10-4 Information Console help link for login page

These links in the Information Console emit a URL for the `wwhelp.html` file and append two parameters to that URL, `context` and `topic`. The URL looks like following example:

```
http://host:8700/iportal/help/wwhelp.htm#context=UserConsole
&topic=Document_list
```

- `host` is the name of the web server serving online help.
- `8700` is the port number for the web and http service.
- `iportal/help/wwhelp.htm` is the path to the help control file.
- `context=UserConsole` is the context parameter that specifies the document root for the required help collection. This parameter's value is the context for Information Console help, `UserConsole`, and directs the request to the Information Console help collection. The context value is determined by the Information Console application.
- `topic=Document_list` is the topic parameter that locates the required help page. This parameter's value is the topic for viewing and navigating the documents and folders page, `Document_list`, which is mapped to an anchor in the `iPmanaging-reports.3.9.html` file. The topic value is determined by the Information Console application.

Understanding locale support

Actuate provides help in US English. The documentation installer places this help in `<context root>\help`. The installer creates directories for all available locales within `<context root>\help`. The locale directory names are the locale code of the form `<ll_cc>` where `ll` is a language code and `cc` is a country code. The directory

names are all in lowercase letters. Each locale directory contains a `wwhelp.htm` file and directories for each help collection listed in Table 10-2, as shown in Figure 10-5 for the `ac_is` locale.

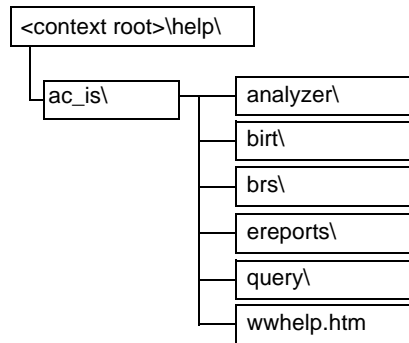


Figure 10-5 `ac_is` locale directory structure

The `wwhelp.htm` files in each locale directory and its collection directories redirect to the files directly in `<context_root>\help`. To support localized online help, place localized files in the appropriate locale directory and modify the `wwhelp.htm` files to not redirect to `<context_root>\help`.

Using a custom help location

Any help system hosted by a web server can provide online help for an Information Console system. To make an external help system available to the Information Console application, the `wwhelp.html` file must redirect help requests to that external system. Any specific help target can link to any specific page.

To redirect help requests from Information Console to an alternate URL, edit or replace the `wwhelp.html` file in `<context root>\help`. You can further specify different targets using the context and topic parameters in the URLs emitted by Information Console in help requests.

Customizing the help location with `wwhelp.htm`

Use the following procedure to create a `wwhelp.htm` file that redirects Information Console context-sensitive help requests to another URL.

- 1 In a text editor, open a new document.
- 2 Write the required pieces of an HTML file, as shown in the following code:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
```

```

<script type="text/javascript" language="JavaScript1.2">
  <!--
  ...
  // -->
  </script>
</head>
<body>
</body>
</html>

```

- 3 Within the script block, write the javascript method `GetParameter` to capture URL parameters, as shown in the following code:

```

// get parameters from the URL
//
method GetParameter( name )
{
  var regexS = "[\\?&]+" + name + "=( [^&#]*) ";
  var regex = new RegExp( regexS );
  var results = regex.exec( window.location.href );
  if( results == null )
    return "";
  else
    return results[1];
}

```

- 4 As shown in the following code, create a method to perform the following tasks:

- Operate the page.
- Use `GetParameter` to obtain the topic and context from the URL.
- Open a URL based upon the topic and context.

```

method LaunchHelp()
{
  // Get URL parameters
  var context = GetParameter( 'context' );
  var topic = GetParameter( 'topic' );

  var baseURL = "http://myhelpserver/viewer/wwhelp.htm";

  // Begin flow control using context
  switch (context)
  {
    // map the "BIRTIV" context to an outside URL
    case "BIRTIV" :
      self.location = baseURL + "?single=true&context=" +
        context + "&topic=" + topic ;
      break;
  }
}

```

```

// map the "UserConsole" context to an outside URL
case "userconsole" :
    baseURL = "http://myhelpserver/iPortal/wwhelp.htm";
    self.location = baseURL + "?single=true&context=" +
context + "&topic=" + topic ;
    break;

//the default behavior
default :
    self.location = baseURL ;
}
}

```

The LaunchHelp() method gets the context and topic information from the URL with two calls to GetParameter. The baseURL is set to the myhelpserver application's online help. The flow control switch statements activate specific URLs depending upon the context. Because the myhelpserver application uses the same context and topic variables as standard Information Console help, they are used directly in constructing the URL when activating the self.location methods.

- 5 Replace the <body> tag with the body tag in the following line:

```
<body onLoad="LaunchHelp();" >
```

The onLoad parameter activates LaunchHelp() when the page loads.

- 6 Save the file as wwhelp.htm in the <context root>\help directory.
- 7 Test the results by opening Information Console and selecting a help link. The resulting page is from the custom application. For example, the help link on the login page pictured in Figure 10-4 would link to http://myhelpserver/iportal/help/wwhelp.htm?single=true&context=UserConsole&topic=Login_MyDoc_Enterprise.

Creating a localized help collection

Actuate Information Console supports localizing help collections by placing localized help files into the help directory for the appropriate locale. The <context_root>\help directory contains several locale-specific help directories. For example, the United States English help subdirectory is <context_root>\help\en_us. Other help locale directories can be populated with localized help to provide help for customers in other locales and in other languages. In order to maintain proper help navigation and context-sensitive help links, localized help pages must have the same name as the help pages provided by Actuate.

How to create a localized help collection

Use the following procedure to create a localized online help collection for Information Console that maintains context-sensitive help requests and help navigation.

- 1 Copy all of the non-locale-specific directories from `<context_root>\help` into the appropriate locale-specific directory. For example, for the Italian locale, copy the files into `<context_root>\help\it_it`.
- 2 Create localized versions of existing help files in a separate directory.
- 3 In the locale-specific directory, copy the localized versions of the help files over the English files of the same name. The localized help can be accessed using the following URL:

```
http://localhost:8700/iportal/help/<locale-specific directory>/wwhelp.htm
```

For example, for the Italian locale-specific help, use the following URL:

```
http://localhost:8700/iportal/help/it_it/wwhelp.htm
```

- 4 Test the results by opening Information Console, selecting the new locale on the login page, and selecting a help link. The resulting page is from the custom application. For example, the help link on the login page shown in Figure 10-4 would link to `http://localhost:8700/iportal/help/it_it/wwhelp/wwhimpl/common/html/wwhelp.htm#href=using-ip/iPmanaging-reports.3.02.html#229645&single=true`.

How to make locale-specific online help the default help

Use the following procedure to make a locale-specific help collection the default help for Information Console.

- 1 Open `wwhelp.htm` in the `<context root>\help` directory in a text editor. Find the following line:

```
setTimeout("location.replace(\"/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

Add the locale-specific directory to the URL string, as shown in the following code:

```
setTimeout("location.replace(\"/<locale-specific directory>/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

For example, to set the Italian locale as the default locale for context-sensitive help, change the line to the following one:

```
setTimeout("location.replace(\"/it_it/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

- 2 Save and close `wwhelp.htm`.

- 3 Copy the all of the non-locale-specific directories from <context_root>\help into each English locale-specific directory - en_au, en_bz, en_ca, en_gb, en_ie, en_nz, en_us, and en_za. For example, for US English, copy the files into <context_root>\help\en_us.

- 4 In each english locale-specific directory, open wwhelp.htm in a text editor. Find the following line:

```
setTimeout("location.replace(\"../wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\")
```

Add the locale-specific directory to the URL string, as shown in the following code:

```
setTimeout("location.replace(\"/<locale-specific directory>/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

For example, to set US English help to the en_us locale for context-sensitive help, change the line to the following one:

```
setTimeout("location.replace(\"/en_us/wwhelp/wwhimpl/common/html/switch.htm\" + Parameters + "\");", 1);
```

- 5 Test the results by opening Information Console and selecting a help link. The resulting page is from the custom application. For example, the help link on the login page shown in Figure 10-4 would link to http://localhost:8700/iportal/help/it_it/wwhelp/wwhimpl/common/html/wwhelp.htm#href=using-ip/iPmanaging-reports.3.02.html#229645&single=true.

Then, test an English locale by selecting an English locale on the login page and then selecting a help link. The resulting page is from the English locale help. For example, the help link on the login page shown in Figure 10-4 would link to http://localhost:8700/iportal/help/en_us/wwhelp/wwhimpl/common/html/wwhelp.htm#href=using-ip/iPmanaging-reports.3.02.html#229645&single=true for the US English locale.

Customizing icons, links, and the company logo

The online help pages organize navigation and content into frames. To change the fonts, colors, and icons of customized help, change each frame's content or style file individually.

Changing the corporate logo

The corporate logo is displayed in the content frame based on the image tags in the content pages. Figure 10-3 shows the title page for the help system. This page contains a large logo image. Individual content pages contain a small logo in the footer, as shown in Figure 10-6. To change this logo, change the image tag on

every content page.

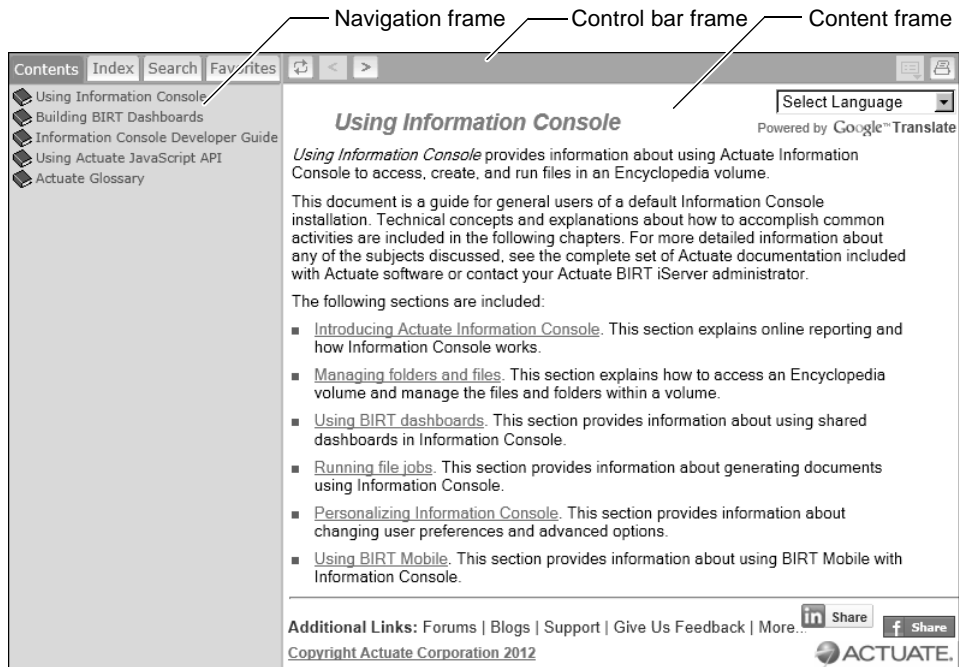


Figure 10-6 Help frames

Changing the corporate logo on the title page

Because this title page is not directly tied to a document, the title page does not reside in an individual document root. The path of the title page, default.htm, is:

```
<context root>\help\wwhelp\wwhimpl\common\html\default.htm
```

Changing the image tag for the logo in this file changes the logo on the title page.

How to change the logo on the title page

Use the following procedure to change the company logo that is displayed on the help title page in the content frame.

- 1 Copy your corporate logo image file into the <context root>\help\wwhelp\images directory.
- 2 In a text editor, open the <context root>\help\wwhelp\wwhimpl\common\html\default.htm file.

3 Locate the following block of code:

```
<!-- Table for floating ActuAteLogo -->
<Table Cols="1" Border="0" cellpadding="0" cellspacing="0"
  height="80%" width="100%">
  <tr>
    <td height="300" valign="bottom" >
      <P align="right">
        <IMG Alt="Actuate Corporation"
          src="../../images/actuate_logo.gif"><br />
        <br />
      </P>
    </td>
  </tr>
</Table>
```

4 Change actuate_logo.gif to the name of your corporate logo image file.

5 Change the value of the Alt parameter to your company name.

6 Save and close default.htm.

Changing the logo in help content pages

The footers in the content pages display the Actuate corporate logo by default. To change the corporate logo displayed on a content page, alter the HTML markup to use a different logo. Actuate uses the corporate logo as a link to the Actuate corporate web application. Change this link so that the image is a link to your corporate web application.

How to change the corporate logo on a help content page

Use the following procedure to alter the corporate logo and corporate web application link in a content page.

1 Copy your corporate logo image file into the <document root>\images directory for the help topic content you wish to change. For example, to change the logo in the “Using Information Console” help topic, the document root is the <context root>\help\using-ip directory.

2 In a text editor, open the first content page file in the document root. For example, the first content page in the “Using Information Console” documentation is iUsing-intro.2.01.html.

3 Locate the following block of code:

```
<table align="right" border="0" cellspacing="0"
  cellpadding="0">
<tr>
  <td align="right" width="95%" >
    <span style="font-size: 10px ;font-family: Arial,
      Helvetica, sans-serif">
```

```

        <a href="http://www.actuate.com">
        </a>
        <!--
        <a href="mailto:info@actuate.com">info@actuate.com</a>
        -->
        </span>
    </td>
    <td width="5%" />
</tr>
</table>

```

- 4 Change `http://www.actuate.com` to the address of your corporate web application.
- 5 Change `actuate_logo_sm.gif` to the name of your corporate logo image file.
- 6 Change the value of the `Alt` parameter to your company name.
- 7 Change the width and height attributes to display the logo image properly.
- 8 Save and close the content file.
- 9 Repeat steps 2 through 8 for each content file you need to change.

Changing the additional links footer in help content pages

The footers in the content pages also display a series of additional links by default. Actuate uses these additional links to jump to locations in its corporate web site. To change the links displayed on a content page, alter the HTML markup to use different links.

How to change the additional links footer on a help content page

Use the following procedure to alter the additional links footer in a content page.

- 1 In a text editor, open the first content page file in the document root. For example, the first content page in the "Using Information Console" documentation is `iPusing-intro.2.01.html`.
- 2 Locate the following block of code:

```

<table>
<tr>
<td style="color: #003366;font-family:sans-serif;font-
weight:bold;font-size:small;">
    Additional Links:
</td>
<td class="td-link">
    <A HREF="http://www.actuate.com/actuate11/forums"
    target="_blank">Forums |</A>
</td>

```

```

<td class="td-link">
  <A HREF="http://www.actuate.com/actuate11/blog"
    target="_blank">Blogs |</A>
</td>
<td class="td-link">
  <A HREF="http://www.actuate.com/actuate11/esupport"
    title="See what BIRT experts are saying"
    target="_blank">Support |</A>
</td>
<td class="td-link">
  <A HREF="#" onclick="sendEmail();"> Give Us Feedback |</A>
</td>
<td class="td-link">
  <A HREF="http://www.actuate.com/actuate11/resources"
    target="_blank"> More...</A>
</td>
</tr>
</table>
<hr />

```

- 3 Change all the links to `http://www.actuate.com/` to addresses for your corporate web application.
- 4 Save and close the content file.
- 5 Repeat steps 2 through 4 for each content file you need to change.

Changing the Google translate element in help content pages

A Google translate element appears in the content pages by default. To change the element displayed on a content page, alter the HTML markup to use different content.

How to change the Google translate element on a help content page

Use the following procedure to alter the additional links footer in a content page.

- 1 In a text editor, open the first content page file in the document root. For example, the first content page in the "Using Information Console" documentation is `iPusing-intro.2.01.html`.
- 2 Locate the following block of code:

```

<div id="google_translate_element" style="text-align:
  right;"></div>
<script>
function googleTranslateElementInit() {
  new google.translate.TranslateElement({
    pageLanguage: 'en'
  }, 'google_translate_element');
}

```

```

}
</script>
<script src="//translate.google.com/translate_a/
    element.js?cb=googleTranslateElementInit"></script>

```

- 3 Delete or replace this code with custom content.
- 4 Save and close the content file.
- 5 Repeat steps 2 through 4 for each content file you need to change.

Changing icons

To change the icons for the controls in the navigation frame and the control bar frame, replace the current image files with different ones. The icon images are located in the <context root>\help\wwhelp\wwhimpl\common\images directory. Replacing these image files changes the icons used for the control bar and navigation frames. Table 10-5 lists and describes the image files for the icons.

Table 10-5 Help content management files



















Image	Filename	Purpose	Location
	bkmrk.gif	Bookmark the current page.	The control bar frame
	bkmrkx.gif	The bookmark method is not available.	The control bar frame
	doc.gif	Open a single file in the table of contents.	The navigation frame
	email.gif	E-mail a link to the current page.	The control bar frame
	emailx.gif	E-mailing a link is not available.	The control bar frame
	fc.gif	Expand a help topic or sub-topic in the table of contents.	The navigation frame
	fo.gif	Collapse a help topic in the table of contents.	The navigation frame
	frameset.gif	Open the control frame.	The control bar frame

Table 10-5 Help content management files

Image	Filename	Purpose	Location
	next.gif	Go to the next page.	The control bar frame
	nextx.gif	There is no next page available.	The control bar frame
	prev.gif	Go to the previous page.	The control bar frame
	prevx.gif	There is no previous page available.	The control bar frame
	print.gif	Print the current page.	The control bar frame
	printx.gif	Printing is not available for this page.	The control bar frame
	related.gif	View related topics.	The control bar frame
	relatedx.gif	The related topics method is not available.	The control bar frame
	sync.gif	Synchronize the frames so that the control frame matches the content frame.	The control bar frame
	syncx.gif	Synchronizing frames is not available.	The control bar frame

Changing the browser window title

To change the title displayed in the browser's title bar when viewing online help, alter the title.js file for each document root. The browser's title bar appears as shown in Figure 10-7.

**Figure 10-7** The browser's title bar

How to change the text displayed in the browser's title bar

Use the following procedure to change the text displayed in the browser's title bar when you access help.

- 1 Navigate to the <document root>\wwhdata\common directory for the help topic you want to customize. For example, to change the text displayed in the browser title bar when you open the "Using Information Console" help topic, the <document root> is the <context root>\using-ip directory.
- 2 In a text editor, open title.js.
- 3 Locate the line in the code that uses the return method. For the "Using Information Console" help topic, it is the following line:

```
return "Using Information Console";
```
- 4 Change the quoted text value to the text you need to display in the browser's title bar.
- 5 Save and close title.js.

Changing help content

Every piece of content in the Actuate Information Console help system is customizable. The possible content changes fall into the following general categories:

- Changing existing help content
- Adding or removing help topics
- Adding and removing content files
- Changing the table of contents
- Changing the index

Changing existing help content

You can modify any of the existing HTML pages of the Information Console help for any help topic to change the information they contain. These HTML files contain specific <a> tags used for internal navigation and context-sensitive help. In general these tags must remain unchanged to maintain context-sensitive help and internal navigation functionality. Table 10-6 lists the tags and their use.

Table 10-6 Help content reserved tags

Tag examples	Purpose
<code></code>	An anchor for a specific place in a file. This tag is used by internal links and context-sensitive links.
<code></code>	Internal link. This tag is an internal link to an anchor. In this example: <ul style="list-style-type: none">■ UserConsole is the context, a reserved help topic label.■ iPgenerating-reports.4.02.html is the file that the link opens.■ #147349 is the text of the anchor tag that the link accesses.

How to modify the content of existing pages

Use the following procedure to change the help content.

- 1 Navigate to the document root directory for the help topic you want to change. For example, to change the content of a page in the “Using Information Console” help topic, the document root is the `<context root>\using-ip` directory.
- 2 In a text editor, open the content page you need to change. For example, to change the content of the “Chapter 2 Managing folders and files” page, open the `iPmanaging-reports.3.01.html` file.
- 3 Modify the text, being careful not remove any `<a>` tags that provide internal links and context-sensitive links.
- 4 Save and close the content file.

Adding or removing help topics

To add or remove help topics from the application help, you delete or create the document root for that help topic. To prevent the navigation pane controls from generating erroneous links to that help topic, you must also alter the help book list, `books.js`, located in the `<context root>\help\wwhelp\wwhimpl\common\private` directory. The `books.js` file also controls the order in which the help topics appear in the table of contents.

How to remove a help topic from the Information Console help system

Use the following procedure to remove a help topic from the Information Console help system.

- 1 Navigate to the `<context root>\help\wwhelp\wwhimpl\common\private` directory.

2 In a text editor, open the books.js file.

3 Find the following code:

```
function WWHBookGroups_Books(ParamTop)
{
    ParamTop.fAddDirectory("using-ip", null, null, null, null);
    ParamTop.fAddDirectory("dashboards", null, null, null, null);
    ParamTop.fAddDirectory("customizing-ip", null, null, null,
    null);
    ParamTop.fAddDirectory("javascriptapi", null, null, null,
    null);
    ParamTop.fAddDirectory("glossary", null, null, null, null);
}
```

4 Delete the line that adds the directory for the topic that you need to remove.

5 Save and close the books.js file.

6 In the file system, delete the document root for the topic that you removed in step 4.

Adding and removing content files

Individual content files are added or removed from the document root for each top-level help topic. To make the content file available for linking and viewing from the help system, you must also alter the file list, files.js, located at <document root>\wwhdata\common. The files.js file also controls the order of the files in the array for reference by other files. For example, the content of files.js for the using-ip document root looks like the following code:

```
function WWHBookData_Files(P)
{
    P.fA("Using Information Console", "about-ipreports.html");
    P.fA("Introducing Actuate Information Console", "iPusing-
    intro.2.01.html");
    P.fA("Using Actuate Information Console for document delivery and
    collaboration", "iPusing-intro.2.02.html");
    P.fA("About Information Console", "iPusing-intro.2.03.html");
    ...}
```

This code establishes the following structure:

- Each file, about-ipreports.html, iPusing-intro.2.01.html, iPusing-intro.2.02.html, and iPusing-intro.2.03.html, is available for linking and display by Information Console help.
- The first file in the array is about-ipreports.html, which is referenced by the array number 0. The second file in the array is iPusing-intro.2.01.html and is referenced by the array number 1 and so on.

The order of the files in the array always begins with and proceeds from 0. The file array is an internal mechanism that supports referencing these files by number within the help topic.

How to add a content file to the Information Console help system

Use the following procedure to add a content file to the Information Console help system.

- 1 Copy your content file into the document root directory for the help topic you need to enhance. For example, to add a new file to the "Using Information Console" help topic, the document root is the <context root>\using-ip directory.
- 2 Navigate to the <document root>\wwhdata\common directory.
- 3 In a text editor, open the files.js file.
- 4 Find the following code:

```
function WWHBookData_Files(P)
{
  P.fA("Using Information Console", "about-ipreports.html");
  P.fA("Introducing Actuate Information Console", "iPusing-
    intro.2.01.html");
  P.fA("Using Actuate Information Console for document delivery
    and collaboration", "iPusing-intro.2.02.html");
}
```

- 5 Add a P.fA(...); entry for the file to add, placing it where you need it to appear in the file array relative to the other entries.

P.fA(...); requires two parameters. The first is a string that describes the file. The second is the name of the file. Both parameter values must individually be within quotation marks and separated by a comma.

- 6 Change the parameter values for the other P.fA(...) calls as needed.
- 7 Save and close files.js.

Changing the table of contents

Help topics are established in the table of contents by the title.js file in the <document root>\wwhdata\js\ directory for each help topic. For example, the title.js file for the using-ip document root looks like the following code:

```
method WWHBookData_Title()
{
  return "Using Information Console";
}
```

This code indicates that the table of contents text for this help topic is "Using Information Console." Figure 10-8 shows the hierarchy produced by the code above.

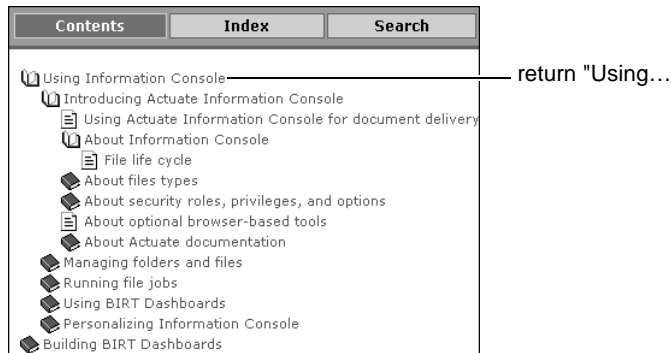


Figure 10-8 The help topic entry in the table of contents

The table of contents displays nested help topics as listed in the toc.js file located in the <document root>\wwhdata\js directory. The toc.js file also controls the following items:

- The table of contents hierarchy
- The text that appears in the table of contents
- The file that opens when a user selects a table of contents entry

For example, part of table of contents entry for the Using Information Console chapter in the toc.js file for the using-ip document root looks like the following code:

```
var A=P.fN("Introducing Actuate Information Console","1");
var B=A.fN("Using Actuate Information Console for document
    delivery and collaboration","2");
B=A.fN("About Information Console","3");
var C=B.fN("File life cycle","4");
B=A.fN("About files types","5");
C=B.fN("File operations","6");
C=B.fN("File categories","7");
var D=C.fN("Folders","7#161885");
D=C.fN("Information Objects You Can Query","7#176010");
D=C.fN("Queries","7#169757");
D=C.fN("Documents You Can View","7#169923");
D=C.fN("Items You Can Run","7#154915");
```

This code establishes the following structure:

- The top-level entry, A, is file "1". File 1 is in position 1 of the internal file array established by files.js. For example, in the using-ip document root, this file is iPusing-intro.2.01.html.
- Entries are created to reside in the next level under the top-level entry using the variable B. Entries in the third level of the table of contents are created using the variable C, and in the fourth level using the variable D. The entries

link to file or anchors within a file referenced by the internal file array number. For example, “7#161885” links to the anchor in file “7” of the file array, iPusing-intro.2.07.html.

- The text that appears in the table of contents for each entry is explicitly defined. For example, the text for the top-level entry is “Using Information Console”.

Figure 10-9 shows the hierarchy produced by this code.

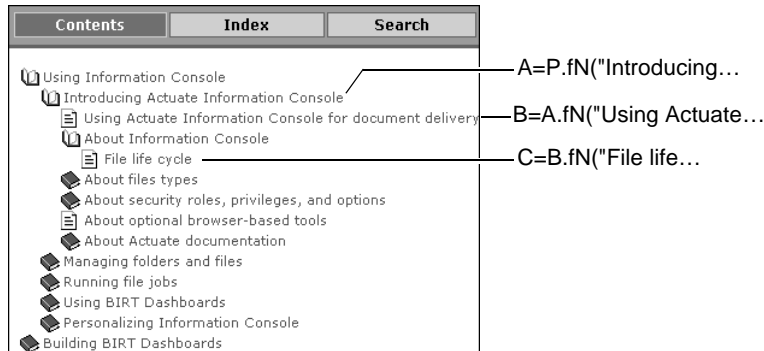


Figure 10-9 The table of contents hierarchy for using-ip

How to add a content file link to the table of contents hierarchy

Use the following procedure to add a content file link to the table of contents hierarchy for the Information Console help system.

- 1 If you are linking to an anchor, navigate to the document root directory. Open the content file that contains the anchor to which the table of contents will link. Determine the value of the name attribute for the anchor. Then, close the content file without saving it.
- 2 Navigate to the <document root>\wwhdata\common directory.
- 3 In a text editor, open the files.js file and determine the internal file array number for the content file, either that you opened in step 1 or that you are linking to directly. Close files.js without saving it.
- 4 Navigate to the <document root>\wwhdata\js directory.
- 5 In a text editor, open the toc.js file.
- 6 Add an entry to toc.js for the table of contents entry using the following format:

```
var B=A.fN("About business reporting using Actuate
products", "1#147349") ;
```

- var is a keyword that must precede the entry if B has not been defined as a variable in this file prior to this line. Do not use var if B has already been defined.
- B is the table of contents hierarchy level of the new table of contents entry.
- A is the table of contents hierarchy level above the level of the new table of contents entry.
- "About business reporting using Actuate Products" is the string to display in the table of contents for this entry.
- 1 is the array number of the target file established in step 3.
- #147349 is a number sign (#) followed by the value of the name attribute for the anchor established in step 1, if it is applicable. Do not append any additional characters to the array number of the target file if you are just linking to the file and not to a marker.

7 Save and close toc.js.

Changing the index

The index displays keywords for help topics from individual content files. The index.js file located in the <document root>\wwhdata\js directory contains the index entries. The index.js file controls the following items:

- The index hierarchy
- The text that makes up the index entries
- The content to which the index entries link

For example, in the using-ip document root, the index entry for QBE expressions, starting at the letter Q, looks like the following code:

```
A=P.fA("Q",null,null,"002");
B=A.fA("QBE expressions");
C=B.fA("adding date values to",new Array("68#439828"));
C=B.fA("creating",new Array("66#554566","68#553349"));
C=B.fA("defining ad hoc parameters and",new
    Array("66#439403","67#439717"));
C=B.fA("entering literal characters in",new
    Array("68#553289","68#515853"));
C=B.fA("matching string values and",new Array("68#514920"));
C=B.fA("retrieving blank characters and",new Array("68#515837"));
C=B.fA("retrieving null values and",new Array("68#439781"));
```

This code establishes the following structure:

- The top-level entry, A=P.fA, is the label "Q". This entry links to the "002" frame, which is the navigation frame.

- The first entry below “Q” is the “QBE expressions” entry. This entry is one level down in the hierarchy, B=A.fA, of the index for “Q”. “QBE Expressions” is merely a label and does not link to anything.
- On the next level down in the hierarchy, C=B.fA, has seven entries, one for each of the sub-topics of QBE Expressions. Each entry has a label and an array of links to topics that the user can choose.

Figure 10-10 shows the hierarchy produced by this code.

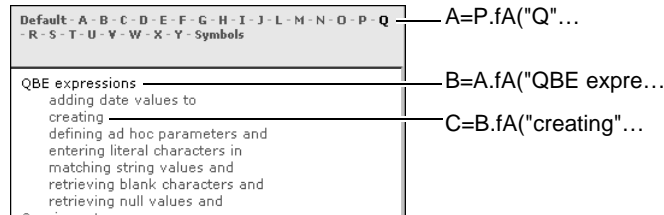


Figure 10-10 The index hierarchy for using-ip

How to add a marker link to the index hierarchy

Use the following procedure to add a marker link to the index hierarchy of the Information Console help system.

- 1 Navigate to the document root directory. Open the content file that contains the anchor to which the index entry will link. Determine the value of the name attribute for the anchor. Then, close the content file without saving it.
- 2 Navigate to the <document root>\wwhdata\common directory.
- 3 In a text editor, open the files.js file and determine the internal file array number for the content file that you opened in step 1. Close files.js without saving it.
- 4 Navigate to the <document root>\wwhdata\js directory.
- 5 In a text editor, open the index.js file.
- 6 Add an entry to index.js for the index entry and anchor link using the following format:

```
var B=A.fA("QBE expressions",new Array("3#394929"));
```

- var is a keyword that must precede the entry if B has not been defined as a variable in this file prior to this line. Do not use var if B has already been defined.
- B is the index hierarchy level of the new index entry.
- A is the index hierarchy level above the level of the new index entry.
- "QBE expressions" is the string to display in the index for this entry.
- 3 is the array number of the target file established in step 3.

- #394929 is a number sign (#) followed by the value of the name attribute for the anchor established in step 1.

To link the index entry to more than one marker, add each marker link to the list within the new Array parameters. Enclose each anchor reference in quotation marks. Delimit the anchor references with commas, shown in the following example:

```
var B=A.fA("QBE expressions",  
    new Array("#394929", "#394380", "#394677"));
```

- 7** Save and close index.js.

Index

Symbols

- _ (underscore) character 13
- ; (semicolon) character 13
- : (colon) character 12
- ! (exclamation point) character 13
- ? (question mark) character 13
- . (period) character 13
- , (comma) character 13
- " (double quotation mark) character 13
- { } (curly brace) characters 159
- @ (at-sign) character 12
- * (asterisk) character 12, 148
- / (forward slash) character 13
- \ (backslash) character 12
- & (ampersand) character 12
- # (number sign) character 13
- % (percent) character 13
- + (plus sign) character 13
- < (less than) character 13
- = (equals sign) character 13
- > (greater than) character 13
- \$ (dollar sign) character 13

A

- about page 98, 114
- access rights. *See* privileges
- accessing
 - Analytics Cube Viewer 160
 - application servers 52
 - cascading style sheets 52, 55
 - channels 10, 84
 - dashboards 10
 - Encyclopedia volumes 89, 235
 - help content pages 253, 255
 - home page 131
 - Information Console functionality 11, 50, 74, 82, 176, 224
 - Interactive Viewer 17
 - JavaScript files 52, 55
 - JSPs 32
 - report files 226
 - repository items 10, 84, 137

- resource bundles 197, 211
- resources 18, 55, 80
- session-specific information 51
- skin manager 59
- tag libraries 52
- templates 53
- web applications 35, 132
- accessToGrant parameter 155, 180
- AcChannelFilter variable 117
- accordion.css 67
- AcCreateQueryAction class 225
- AcFilesFoldersFilter variable 138
- AcFilesFoldersTypeFilter variable 138
- AcGetFileDetailsAction class 226
- AcGetFolderItemsAction bean 50
- AcGetJobDetailsAction class 227
- aciportal.jar 224
- AcLastViapplicationdFolder variable 138
- AcRunQueryAction class 225
- AcServlet class 176
- actabpanel tag library 194, 195
- Action class 50, 224
- action forms 224, 227
- action forms classes. *See* forms package
- action parameter 147
- action path names 57
- action path specifications 50
- action paths 30, 50, 51, 57
 - See also* URIs
- action tag 50
- ActionForm class 224
- actions 17, 30, 85, 98, 102
- actionServlet component 30
- activePortal directory 32, 33
- ActivePortalResources.properties 18
- activity logs 77, 78
- Actuate eReportlets 206
- Actuate eReports
 - See also* reports
- Actuate Query pages. *See* query pages
- Actuate Viewer 161
- actuate_logo_sm.gif 264
- actuate_logo.gif 263

- AcViewCubeAction class 225
- acweb.war 6
- AddFile subfeature 85
- adding
 - action paths 50, 51, 57
 - background images 68
 - context roots 35–37
 - custom JSPs 32, 53
 - custom security adapters 240, 241–243
 - experience levels 20, 21, 26, 93, 94
 - features 18, 20
 - folders 17, 117, 226
 - functionality levels 14, 15, 16, 20
 - help topics 269
 - hyperlinks 226
 - locales 86
 - server profiles 40, 80
 - skins 59, 60, 61, 234
 - table of contents 170, 208
 - time zones 86
 - upload security adapters 246, 247–249
 - URI parameters 12, 13
 - web pages 10, 51, 52, 56, 88
- addressing e-mail 142
- Administrator functionality level 14, 15, 83
- Advanced experience level 20, 92
- Advanced functionality level 14, 15, 83
- AdvancedData subfeature 17, 85
- ageDays parameter
 - execute query page 125
 - execute report page 127
 - ExecuteReport servlet 180
 - submit job page 155
- ageHours parameter
 - execute query page 125
 - execute report page 127
 - ExecuteReport servlet 180
 - submit job page 155
- aging intervals 125, 127, 155, 180
 - See also* archive policies
- allscripts.js 172
- allstyles.css 65
- ANALYSIS formats 178
- ANALYSIS value 213
- Analytics Cube Viewer
 - configuring 90–94
 - defining experience levels for. *See* experience levels
 - disabling features for 20, 21, 93
 - opening 160
 - viewing cube reports and 160
- Analytics Option 20, 90
- ANALYTICS_BASE_EXPLEVEL_NAME
 - parameter 91
- ANALYTICS_CUBE_VIEW_RECORDS
 - parameter 91
- ANALYTICS_CUBE_VIEWER_HEIGHT
 - parameter 91
- ANALYTICS_CUBE_VIEWER_WIDTH
 - parameter 91
- ANALYTICS_ENABLE_ONETIME_DOWNLOAD
 - parameter 91
- ANALYTICS_ENABLE_SAVE_VIEW
 - parameter 91
- analyticsbrowsefolder action 104
- AnalyticsExperienceLevel tag 86, 92, 94
- Apache Jakarta Struts. *See* Jakarta Struts
- Apache Tomcat engine 35
- Apache Tomcat service 37
- application context roots 11, 31, 51
- application servers 5, 7, 35
 - See also* servers
- applications
 - accessing 35, 132
 - building user interface for 4, 53, 57, 194
 - changing 6, 37, 49, 57
 - configuring 47, 74
 - creating context root for 35–37
 - creating page-specific content for 56
 - customizing pages for. *See* web pages
 - deploying 238
 - designing custom reporting 8, 30, 37, 48, 57
 - determining state of 52
 - getting session information for 51
 - grouping 34
 - linking help files to 255, 263, 264, 269, 270
 - previewing skins for 60, 61
 - setting default locale for 76, 86
 - setting default time zone for 76
 - setting global styles for 57–68
 - testing 35
 - translating. *See* locales

- applyFilter parameter 134, 138
- archive policies 126, 127, 156, 180
- archiveBeforeDelete parameter
 - execute query page 125
 - execute report page 127
 - ExecuteReport servlet 180
 - submit job page 156
- archivePolicy parameter
 - execute query page 126
 - execute report page 127
 - ExecuteReport servlet 180
 - submit job page 156
- archiving
 - query output 125
 - report files 127, 156, 180
- array.js 172
- arrays 199, 215
- ASCII formats 87
- attachments 17, 157
- authenticate method 239, 241, 243
- authenticate page 98, 114
- authenticate.jsp 115
- authentication
 - accessing applications and 238
 - accessing corporate networks and 238
 - customizing 238, 241
 - issuing URIs and 101
 - logging in to Information Console and 239, 240
 - starting user sessions and 114
- authentication IDs 52, 101, 210, 229, 232
- authentication information 7, 114
- authexpired action 103
- authID variable 211
- autoarchive policies 126, 127, 156, 180
- autoarchiving. *See* archiving

B

- background images 68
- background.gif 68
- backward compatibility 32
- banner
 - adding links to 82
 - changing welcome text in 49
 - displaying 115, 139
 - hiding features in 16
 - replacing images in 68
 - specifying functionality levels and 15
- banner elements 133, 139
- banner labels 67
- banner page 98, 115
- banner styles 67
- banner.jsp 115
- BaseActionForm class 227
- BasedOnFileID parameter 131
- Basic functionality level 14, 15, 83
- beans 52, 53, 66, 224, 232
- beans package 224, 229
- binary files 176
- BIRT Interactive Viewer. *See* Interactive Viewer
- BIRT iServer System. *See* iServer System
- BIRT iServer. *See* iServer
- BIRT libraries 33
- BIRT report documents 186
 - See also* BIRT reports
- BIRT Report Studio 67
- BIRT reports 94, 95, 161
 - See also* reports
- BIRT servlet 161
- BIRT Studio 57, 94
- BIRT Studio link 131
- BIRT Viewer 94, 95, 161
- BIRT_RENDER_FORMAT_EMITTER_ID_MAPPING parameter 75
- body element 49
- bookmark parameter 186
- branding 49, 68
- breadcrumb.jsp 132
- breadcrumbs 56, 131
- browse file page 98, 116
- browse page. *See* browse file page
- browse.jsp 116
- browsefile action 103, 104, 226
- browsefile.do 111
- BrowseFileActionForm class 226
- browsers. *See* web browsers
- browsertype.js 172
- browsing 116, 226
- bundle tag 196
- BundleTag class 197
- bundling resource files 196, 211
- buttons 21, 58

C

- CACHE_CONTROL parameter 75
- caching web pages 49, 75, 90
- calendar page 98, 116
- calendar.jsp 116
- calendarlayer.js 172
- canceljob action 103, 104
- canceljob page. *See* request drop page
- canceljob.do 112
- cancelreport action 104
- cascading style sheets
 - accessing 52, 55
 - changing styles in 65
 - customizing web pages and 48, 68
 - embedding 151, 188
 - linking to JSPs 65, 66
 - specifying color settings in 66
 - updating changes to 50
 - viewing changes to 67
- case sensitivity 30, 98, 155, 176, 194
- categories.jsp 69
- CATEGORY parameter 91
- category parameter 127
- cbFail parameter 134
- cbSuccess parameter 134
- ChangeResponseLocale message 196
- changing
 - action paths 51
 - actions 85
 - background images 69
 - company logos 261–264
 - configurations 37, 38
 - default servers 40
 - Encyclopedia volumes 40
 - error messages 42
 - experience levels 20, 21, 26, 93
 - file names 57
 - font styles 66
 - functionality levels 20
 - help indexes 274
 - help topics 268
 - icons 85, 266
 - images 49, 61, 68–70
 - label key values 19
 - landing pages 48–49
 - link icons 19
 - link targets 19
 - locales 39, 143
 - passwords 143
 - reporting applications 6, 37, 49, 57
 - servlets 176
 - side menu 85
 - style definitions 65, 66
 - templates 54, 55
 - time zones 39
 - user interface elements 70
 - web browser titles 267
 - web pages 55, 56, 88
- channel classes 224
- channel contents list page 136
- channel icons 142
- channel parameter 137
- channelIcons parameter 142
- channelID parameter 120
- ChannelListActionForm class 224
- channelName parameter 120, 121, 134
- channelnoticelist.jsp 136
- channels
 - accessing 10, 84
 - displaying 136, 142, 224
 - filtering 117
 - removing notifications from 119
 - sending notifications to 134, 180
 - setting properties for 116
 - subscribing to 17, 136, 225
 - unsubscribing from 136
 - viewing contents 10, 136
- Channels attribute (features) 16, 19, 84
- channels directory 10
- Channels link 19
- channels list page 136
- channels page 99, 116
- channels parameter 142, 180
- channels.jsp 117
- channelsicon.gif 19
- character encoding 12, 14, 87
 - See also* encoding parameter
- character sets 14
- character substitution 12
- charts 182, 188
- check boxes 137
- class reference (JavaBeans) 224, 229

- classes 48, 49
- Classic skins 58
- clearFilter parameter 134
- cloning skins 60, 61
- closedfoldericon.gif 68
- closeX parameter 167
- closing parent windows 167
- clusters 4, 5, 6, 7, 90
- code 52
- color names 61, 66
- color palettes 61
- colors 57, 64, 66
- column headers 163, 178
- comma-separated formats. *See* CSV formats
- comments 49
- common tag library 194, 195
- company logos 68, 261–264
- compiling JSPs 30
- completed jobs page 134
 - See also* completed request page
- completed request page 99, 117
- completedjob.jsp 117
- completion notices 142
- component tag 196, 198
- componentID parameter
 - GetDynamicData servlet 182
 - GetReportData servlet 183
 - save as page 151
 - view TOC page 170
 - ViewEmbeddedObject servlet 188
 - ViewPage servlet 189
- componentIdentifier tag 196, 198
- componentIdentifierList tag 196, 199
- ComponentIdentifierListTag class 199
- ComponentIdentifierTag class 198
- componentList tag 196, 199
- ComponentListTag class 199
- componentName parameter 183, 189
- ComponentTag class 198
- componentValue parameter 183, 189
- configuration files 38, 74
- configuration parameters 95
 - Analytics Cube Viewer 90
 - changing 38
 - Information Console 38, 74
 - iServer connections 89
- configurations
 - accessing Information Console
 - functionality and 50, 74, 82
 - adding web pages and 51
 - changing messages and 42, 87
 - connecting to iServer and 80, 89
 - creating custom applications and 37–47, 74
 - customizing context root and 35
 - defining experience levels and 20, 21, 26, 86, 91–94
 - defining features and 16, 18, 84
 - defining functionality levels and 14, 16, 18, 20, 82–86
 - defining subfeatures and 17, 85
 - deploying an upload security adapter and 248
 - deploying IPSE applications and 243
 - disabling load balancing and 7
 - generating locale-specific sites and 86
 - initiating actions and 102
 - invoking servlets and 176
 - redirecting web pages and 9
 - renaming files and 57
 - running Analytics Cube Viewer and 90, 91
 - running BIRT reports and 94
 - running multiple applications and 7
 - setting up firewalls and 8, 9, 239
 - specifying time zones and 86
 - updating changes to 37
- confirmation messages 53
- confirmKey parameter 142
- connection information 74
- connection parameters 30
- connection pools 89
- CONNECTION_TIMEOUT parameter 76
- connectionHandle parameter
 - GetDynamicData servlet 182
 - GetStaticData servlet 185
 - print page 149
 - search report page 163
 - view default page 165
 - view frame set page 167
 - view TOC page 170
- connections
 - accessing Encyclopedia and 30, 102, 148, 245

- connections (*continued*)
 - accessing private networks and 239
 - dropping 76
 - establishing iServer 89, 210, 239
 - protecting data and 238–239
 - setting maximum number of 89
 - timing out 78
- Content class 200
- content element 56
- content tag 195, 200
- context menus 23, 24
- context roots 11, 31, 35, 51
- context-sensitive help 255
- controlName parameter 179
- converter 183, 189, 205
- converter.js 172
- converterParam parameter 183, 189
- COOKIE_DOMAIN parameter 76
- COOKIE_ENABLED parameter 76
- COOKIE_SECURE parameter 76
- cookie.js 172
- cookies 76, 101, 137
- copyFileFolder tag 195, 200
- CopyFileFolderTag class 201
- copying
 - folders 200
 - image files 69
 - query statements 159
 - report files 200
 - skins 60, 61
- corporate logos. *See* company logos
- counting report pages 203
- country codes 86, 87
- create folder page 99, 117
- create query page 99, 118
- create.do 111
- createfolder action 105, 226
- CreateFolder subfeature 17, 85
- createfolder.do 111
- createfolder.jsp 117
- createpage.jsp 118
- createquery action 103, 225
- CreateQueryBean class 225
- creating
 - action paths 50, 51, 57
 - context roots 35–37
 - custom JSPs 32, 53
 - custom security adapters 240, 241–243
 - experience levels 20, 21, 26, 93, 94
 - features 18, 20
 - folders 17, 117, 226
 - functionality levels 14, 15, 16, 20
 - help files 257, 260
 - help indexes 275
 - hyperlinks 226
 - queries 118, 225
 - skins 59, 60, 61, 234
 - table of contents 170, 208
 - upload security adapters 246, 247–249
 - URIs 11, 12
 - user interfaces 4, 53, 57, 194
 - WAR files 5, 6
 - web applications 8, 30, 35, 47–57
 - web pages 10, 52, 56, 88
- credentials. *See* login credentials
- CSS files 48, 52, 65, 67
 - See also* cascading style sheets
- CSV formats 178, 179, 213
- cube classes 225
- cube reports 20, 90, 160
- Cube Viewer. *See* Analytics Cube Viewer
- cube views 91
- cubedetail action 105
- CubeParam class 225
- cubes 128, 160, 225
- custom tags. *See* JSP custom tag reference
- Customization attribute (features) 16, 84
- Customization link 15, 19
- customize action 105
- customize.do 19
- customizing
 - background images 69
 - experience levels 21–26, 92, 94
 - features 18–19
 - file verification 247
 - functionality levels 16–18
 - Information Console 5, 6, 38, 57
 - JSPs 32, 52, 54, 55
 - landing page 48–49
 - messages 42–47, 87
 - online help 252–276
 - reporting applications 8, 30, 35, 47–57
 - security adapters 240–243
 - skins 57, 59, 60, 62, 84, 228

- upload security adapters 246–249
- user authentication 238, 241
- user logins 238
- web pages 48, 55, 56, 88

D

- daemonURL parameter 140
- dashboard directory 10
- dashboard files 32, 53
- dashboard page 99
- dashboard template 53
- DashboardBusinessUser subfeature 85
- DashboardDeveloper subfeature 85
- dashboards 85, 89, 140
 - accessing 10
 - displaying 10
- dashboardtemplate.jsp 53
- data
 - displaying 30, 56, 182
 - embedding in reports 187
 - exporting 150
 - filtering 225, 226, 227, 234
 - getting static 185
 - protecting 238
 - restricting access to 238
 - retrieving 182, 189, 206
 - synchronizing 17
- data analyzer 32
- data cubes 128, 160, 225
 - See also* cube reports
- data filters. *See* filters
- data models 30
- data object executable files 118, 124, 160, 225
 - See also* executable files
- data object values files 118, 124, 160
- database servers 8
- date formats 201
- date patterns 201
- date stamps 126, 129
- dateToDelete parameter
 - execute query page 126
 - execute report page 127
 - ExecuteReport servlet 180
 - submit job page 156
- debug pages 197
- debugging messages 77
- decimal values 66
- default authentication 238
- default banner 115
- default context root 31
- default encoding 14
- default Encyclopedia volume 40, 233
- default error codes 87
- default experience level 26, 93, 232
- default file names 57
- default functionality levels 82
- default images 68
- default locale 18, 39, 76, 86, 232
- default settings 39, 40
- default skins 58, 60
- default time zone 76
- DEFAULT_ESS_VIEWING_FORMAT
 - parameter 76
- DEFAULT_EXPERIENCE_LEVEL tag 93
- DEFAULT_LOCALE parameter 39, 76
- DEFAULT_PAGE_BREAK_INTERVAL
 - parameter 76
- DEFAULT_TIMEZONE parameter 39, 76
- delete file status page 99, 119
- delete job page 99, 119
- delete status page 99, 119
- deletefile action 103, 105
- deletefile page. *See* file drop page
- DeleteFile subfeature 17, 85
- deletefile.do 112
- deletefilestatus.jsp 119
- deletefilestatuspage. *See* delete file status page
- deletefolder page. *See* folder drop page
- DeleteFolder subfeature 17, 85
- deletejob action 103, 105
- deletejob page. *See* delete job page
- deletejob.do 111
- deletejobnotice action 103, 105
- deletejobnotice page. *See* delete status page
- deletejobnotice.do 112
- deleting
 - archived reports 126, 129, 159, 181
 - folders 17, 122
 - help topics 269
 - jobs 119, 123
 - leading/trailing spaces 199
 - notifications 119

- deleting (*continued*)
 - query output 126
 - report files 17, 119, 122
 - skins 60
- delimited text files 164
- demand paging. *See* immediate jobs; progressive viewing
- deploying
 - Information Console 4, 5, 6
 - reporting applications 238
 - reports 5
- depth parameter 170
- designing custom web applications 8, 30, 37, 48, 57
- destination parameter 147
- detail pages 69, 99, 112, 120
- detailicon.gif 69
- details icon 69
- developers 4
- DHTML format 183, 190
- DHTML Reportlets 204, 206
- DHTML reports 155
- DHTML Viewer 114, 124, 168
- DHTMLLong format 183, 190
- DHTMLRaw format 183, 190
- diagnostic information 146
- dialog.css 67
- dialogbase.css 68
- dialogs
 - defining tabs for 215, 218, 219, 221
 - setting orientation of 220
 - setting tab sequence for 216, 217, 218
 - specifying content for 200, 219
- directories 31, 36, 253
- directory names 30
- directory paths. *See* paths
- disk space 79, 148
- display names 86, 87, 93
- DISPLAY value 213
- DISPLAY_NAME tag 93
- displaying
 - banners 115, 139
 - color settings 64
 - completed jobs 117
 - cube reports 160
 - current jobs 150
 - data 30, 56, 182
 - embedded objects 187, 188
 - error messages 124, 145
 - failed jobs 134
 - files and folders list 77
 - folders 139, 234
 - help topics 271, 273
 - images 184, 191
 - locales 86
 - login page 12, 139
 - PDF files 149
 - pending jobs 153
 - report executables 138, 234
 - report parameters 145, 158
 - Reportlets 204
 - reports 10, 11, 32, 79, 94, 95, 186
 - repository information 56
 - search results 57, 131, 162
 - specific report pages 188
 - spreadsheet reports 76
 - subscribed channels 116, 136, 142, 224
 - table of contents 166, 168, 169
 - web pages 50
- DisplayName tag 86, 87
- distributed iServer System. *See* clusters
- do directive 98
- .do files 30
- do_executereport.jsp 179
- do_update page. *See* options page
- docChanFilters parameter 142
- document classes 226
- document files 121, 138, 156, 186, 234
- document type definitions 32
- documentation ix
- documents. *See* reports
- Documents attribute (features) 16, 84
- Documents link 19
- Documents page 56, 226
- doEndTag method 196
- domains 76
- .dov files. *See* data object values files
- Download link 150
- downloadfile action 103
- DownloadFile servlet 177
- DownloadFile subfeature 85
- downloading
 - binary files 176
 - data cubes 160

- reports 150, 177
- search results 163, 178
- DownloadSearchResult servlet 178
- .dox files. *See* data object executable files
- drift.js 172
- drop pages 99, 112, 122
- dynamic data 172, 182, 188

E

- e.Analysis support files 32
- e.reporting server. *See* iServer
- e.Reporting System. *See* iServer System
- e.reports 90, 161
- e.reports. *See* reports; spreadsheet reports
- editors 26
- editTableRow action 105
- email parameter 142
- e-mail. *See* notifications
- embed parameter 185, 188
- embedded objects 151, 187, 188
- embSrvRequester parameter
 - GetReportData servlet 183
 - save as page 151
 - view default page 165
 - view frame set page 167
 - ViewPage servlet 189, 190
- emitters 75
- ENABLE_CLIENT_SIDE_REDIRECT
 - parameter 9, 76
- ENABLE_DEBUG_LOGGING parameter 77
- ENABLE_ERROR_LOGGING parameter 77
- ENABLE_JUL_LOG parameter 77
- enableColumnHeaders parameter 163, 178
- encode method 13
- encoder.js 13, 172
- encoding 12, 14, 87
 - See also* encoding parameter
- encoding parameter 151, 183, 190
- Encyclopedia volume icons 68
- Encyclopedia volumes
 - See also* repositories
 - accessing 89, 235
 - adding objects to 17
 - connecting to 30, 102
 - creating folders for 117, 226
 - deleting objects in 17, 119, 122
 - downloading from 177
 - downloading objects from 17
 - getting current 245
 - getting information about 120, 148
 - running Information Console and 4
 - specifying default 40, 233
 - testing connections to 148, 245
 - writing reports to 129, 159
- engines 79
- environment settings 130
- eReports
 - See also* reports
- erni_config.xml 74
- error action 103
- error codes 45, 87
- error detail page 120
- error levels 45
- error log files 77, 78
- error logging parameter 77
- error messages
 - customizing 42, 45–47
 - displaying 124, 145
 - localizing 87–88
- error page 99, 124
- ERROR_LOG_FILE_ROLLOVER
 - parameter 77
- error.jsp 124
- ErrorMessage.txt 87
- ErrorMessages.properties 45, 87
- ErrorMessages.txt 45
- errors 50, 53, 77
 - See also* error messages
- Excel formats 178, 179
- Excel spreadsheets 150, 156, 184, 190
 - See also* spreadsheet reports
- ExcelData format 156, 184, 190
- ExcelDisplay format 156, 184, 190
- executable files
 - displaying 138, 234
 - generating output for 143, 152, 155
 - running queries and 118, 124, 126, 225
 - selecting 137
 - viewing parameters for 145
- executableName parameter
 - create query page 118
 - execute page 124
 - execute query page 126

- executableName parameter (*continued*)
 - execute report page 128
 - submit job page 156
 - submit page 160
- execute page 99, 124
- execute query page 99, 125
- execute report page 99, 126
- EXECUTE_DASHBOARD_GADGET_GENERATION_WAIT_TIME parameter 77
- EXECUTE_REPORT_WAIT_TIME parameter 77, 124, 127
- executedocument action 103, 106
- executequery action 103
- executequery page. *See* execute query page
- executereport action 103, 106
- executereport page. *See* execute report page
- ExecuteReport servlet 179
- executereport.do 112
- executing
 - Java servlets 176
 - jobs 10
 - queries 124, 125, 159
 - reports 126, 153, 179
- execution requests. *See* jobs; requests
- experience level definitions 93, 94
- experience level names 20
- experience levels
 - adding 20, 26, 93, 94
 - changing 20, 21, 26, 93
 - configuring 85, 91–94
 - customizing 21–26, 92
 - naming 93
 - selecting 18, 20, 28, 91
 - specifying default 26, 93, 232
- EXPERIENCE_LEVEL tag 26, 92
- experience.levels file 20, 26, 74, 91
- expiration intervals 125, 127, 155, 180
 - See also* archive policies
- exporting data 150
- extended character sets 14

F

- Factory service 148
- failComp parameter 142
- failed jobs 134
- failEmail parameter 142

- failover 7
- failure notices 142
- feature definitions 16, 18, 84
- feature IDs 18, 84
- feature lists 230, 233, 234
- feature names 18
- Feature tag 84
- FeatureID tag 16, 18, 84
- FeatureOptionsBean class 228
- features 16, 18–20, 84, 228, 234
- file detail page 121
- file drop page 122
- file IDs 178
- file index page 132
- file list page 137
- file lists 77, 137, 226, 233
- file names 19, 30, 57, 85, 145
- file numbers 78
- file structures 6
- filedetailcontent.jsp 121
- filefolderlist.jsp 132
- filefoldersprivilege action 106, 226
- fileId parameter 178
- FileListActionForm class 226
- filename parameter 148
- files
 - See also* specific type
 - accessing 10, 84, 137, 226
 - archiving 127, 156, 180
 - assigning privileges to 149
 - assigning to template elements 53
 - changing images and 68, 69
 - converting to ASCII 87
 - copying 200
 - creating online help and 253
 - deleting 17, 119, 122
 - displaying 138
 - downloading 177
 - filtering 137
 - generating locale-specific sites and 42
 - getting information about 121, 226
 - linking to 57
 - locating JSP 50
 - logging activities to 77
 - logging errors to 77
 - naming 128, 145, 158, 181
 - overwriting 128, 144, 157, 181

- preserving 157
- renaming 57, 69
- saving 129, 143
- searching 154
- sharing 17
- updating changes to 49
- validating file types for 249
- verifying 250
- FILES_DEFAULT_VIEW parameter 77
- files.js 255
- filesfolders directory 10
- filesfolders tag library 195
- filesfoldersicon.gif 19
- FileUploadActionForm class 228
- filter action forms 225, 226, 227
- filter parameter 134, 138
- filtering
 - channels 117
 - data 225, 226, 227, 234
 - jobs 134
 - report documents 137
- filters 134, 137, 138, 233
- firewalls 8, 9, 238
- floatingfooter parameter 186
- folder detail page 121
- folder drop page 122
- folder icons 68
- folder IDs 118
- folder index page 132
- folder list page 137
- folder lists 77, 137, 226, 232, 233
- folder names 98, 118, 232
- folder parameter 138, 154, 180
- foldericon.gif 68
- folders
 - accessing 10, 84, 137
 - archiving contents 127
 - assigning privileges to 149
 - copying 200
 - creating 17, 117, 226
 - deleting 17, 122
 - displaying 139, 234
 - getting home 210, 230, 245
 - linking to 131
 - navigating through 32, 116
 - searching 84, 154
 - specifying type 145, 233
 - viewing information about 121, 138, 226
- folderType parameter 156
- fonts 57, 64, 66
- footers 186
- FORCED_GC_INTERVAL parameter 77
- forceLogin parameter 101, 239
- format parameter
 - DownloadSearchResult servlet 178
 - GetReportData servlet 183
 - Interactive Viewer servlet 187
 - search report page 163
 - submit job page 156
 - ViewPage servlet 190
- formatDate tag 196, 201
- FormatDateTag class 202
- formats
 - displaying reports and 187
 - executing queries and 126
 - generating output and 156, 183, 190
 - getting 202
 - localizing reports and 201
 - saving reports and 150
- formatting
 - date values 201
 - query output 126
 - search results 163, 178, 213
 - web pages 57
- FormFile class 228
- forms. *See* action forms
- forms classes. *See* forms package
- forms package 224
- forward definitions 51, 102
- forward tag 51
- free disk space 148
- from_page_range parameter 187
- from_page_style parameter 187
- functionality levels
 - adding 14, 15, 16, 20
 - associating with users 14, 16
 - changing 20
 - configuring 82–86
 - customizing 16–18
 - naming 16, 83
 - preserving 20
 - specifying experience levels for 18, 20, 85, 94
 - specifying features for 16, 18, 19, 84

functionality levels (*continued*)
 specifying subfeatures for 17, 85
functionality-level.config 14, 20, 69, 74, 84

G

GADGET_GENERATION_WAITING_TIME
 parameter 78
gadgets 77, 78, 85, 89
garbage collection 77
general options page 99, 130
general.jsp 130
GeneralFilterActionForm class 225, 226
Generate Web Archive option 6
generating
 data cubes 160
 Excel spreadsheets 156, 184, 190
 locale-specific sites 42
 notifications 142
 output 125, 143, 183, 190
 spreadsheet reports 179
 table of contents 170, 208
 WAR files 5, 6
 web pages 10–11, 30, 56
get saved search page 99, 131
getAcLocale method 229
getAdminRights method 229
getAuthid method 229
GetContent method 182
getContextPath method 51
getCurrentfolder method 230
getDefaultAnalyticsExpLevel method 230
GetDynamicData method 188
GetDynamicData servlet 182
getErrorMessage method 247, 249
getExtendedCredentials method 244
getFeatureBean method 228
getFeatureOptionsBean method 230
getFeatures method 230
getfiledetails action 106, 226
getfiledetails page. *See* file detail page
getfiledetails.do 112
GetFileDetailsActionForm class 226
getFilter method 230
getfolderitems action 106
getfolderitems page. *See* folder index page
getfolderitems.do 19, 112, 113

getFormats tag 196, 202
GetFormatsTag class 202
getHeight method 204
getHomefolder method 230
getImage method 228
getIportalid method 52, 230
getjobdetails action 103, 106, 227
getjobdetails page. *See* request detail page
getjobdetails.do 112
getjobdetails.jsp 121
GetJobDetailsActionForm class 227
getjobdetailscontent.jsp 121
getLocale method 230
getMaxJobPriority method 230
getnoticejobdetails action 106
getOnlylatest method 230
getPageCount tag 196, 203
GetPageCountTag class 203
getParameter method 51
getPassword method 230, 241, 244
getportletfolderitems action 106
getProfile method 228, 230
getProperty method 230
getQuery method 225
GetReportData servlet 182
getReportlet tag 196, 204
getReportletData tag 196, 206
GetReportletDataTag class 206
GetReportletTag class 204
getRepositoryType method 230, 244
getRoleNames method 231
getRunAsUser method 244
getsavedsearch action 103
getsavedsearch page. *See* get saved search
 page
getsavedsearch.do 112
getServerUrl method 244
getServerurl method 231
getShowdocuments method 231
getShowexecutables method 231
getShowfolders method 231
getSideBarFeatures method 231
getSidebarSelected method 231
getSkin method 228
getSkinConfig method 231
getSkinName method 231
GetStaticData method 188

- GetStaticData servlet 185
- getStyle method 228
- GetStyleSheet method 188
- getSubfeatures method 231
- getSystemname method 231
- getTimezone method 231
- getTOC tag 196, 208
- GetTOCTag class 208
- getUserAgent method 231
- getUserHomeFolder method 245
- getUserid method 231
- getUserName method 241, 245
- getView method 231
- getVolume method 231, 245
- getWidth method 204
- global style elements 57–68
- goto action 103
- graphical user interfaces. *See* user interfaces
- graphics files. *See* image files
- graphics. *See* images
- graphs. *See* charts
- GroupBean class 227
- groups parameter 180
- GUIs. *See* user interfaces

H

- headers 163, 178
- headline parameter
 - execute report page 128
 - ExecuteReport servlet 180
 - output page 144
 - submit job page 157
- headlines. *See* headline parameter
- help collections
 - See also* help systems
- help content pages
 - accessing 253, 255
 - adding 271
 - changing additional links on 264–266
 - changing company logos on 263–264
 - changing content in 268
 - removing 270
- help directory 253
- help files 252, 257, 260
- help indexes 274, 275
- help keywords 274, 275

- help links 259, 260, 261
- help navigation pages 253, 266
- help systems 257, 268
- help topics 254, 269, 271
- help.js 172
- hexadecimal color values 66
- hexadecimal encoding 12
- HIDEITEM tag 21, 94
- hits parameter 163, 178
- home directory 31
- home folders 131, 210, 230, 233
 - getting 245
- home page 99, 131
- homeFolder parameter 133
- homeFolder variable 211
- homefolder.gif 68
- horizontal bars 21
- hosts 76
- HTML code 48, 55, 57
- HTML files 253
- HTML tables 55, 57
- htmlselect.js 172
- HTTP requests 4, 8, 79, 229
- HTTPS requests 4
- hyperlinks
 - See also* URLs
 - changing targets for 19
 - creating 226
 - defining action paths and 51
 - displaying for specific locales 18
 - embedding 151, 188
 - referencing files and 57
 - retrieving 189
 - setting targets for 19
- hypertext markup language. *See* HTML code

I

- i18n tag library 195, 196
- icon files 19, 68, 69, 85, 266
- icons 266
 - channels 142
 - features 19
 - replacing 61, 68
 - side menu 84
- IContentList interface 224

- ID parameter
 - file or folder drop page 123
 - GetDynamicData servlet 182
 - GetReportData servlet 184
 - GetStaticData servlet 185
 - print page 149
 - search report page 163
 - view default page 165
 - view frame set page 167
 - view TOC page 170
- id parameter 178
- ID tag 84
- id variable 197, 212
- IDAPI. *See* Information Delivery API
- ifExists parameter 125, 144, 157
- image files 64, 68, 228, 266
- image resolution 184
- Image tag 228
- imageid parameter 187
- images
 - adding background 68
 - changing 49, 61, 68–70
 - customizing 68
 - displaying 184, 191
 - embedding 185, 187
 - referencing 69
 - retrieving 185
 - selecting skins and 57
 - uploading 228
- images directory 69
- img tag 261
- immediate jobs 125, 128, 157
- index pages 112, 132
- index.htm file 254
- index.js 255
- information 232
 - See also* data
- Information Console
 - accessing functionality 11, 50, 74, 82, 176, 224
 - adding web pages to 10, 51, 52, 56, 88
 - building user interface for 4, 53, 57, 194
 - changing default settings for 39, 40
 - changing deployed versions of 6
 - changing messages for 42, 45
 - changing side menu for 85
 - changing web pages for 48, 57
 - configuring as web applications 37–47, 74
 - configuring proxy servers for 8, 9
 - creating context root for 35–37
 - creating custom applications for 8, 30, 35, 47–57
 - creating online help for 252–276
 - customizing 5, 6, 57
 - deploying 4, 5, 6
 - displaying information about 114, 135
 - displaying pages for 19, 98
 - grouping applications for 34
 - installing 5, 6, 78
 - localizing messages for 87–88
 - logging in to 12, 139, 240
 - logging out of 140
 - overview 4, 8
 - renaming default files for 57
 - retrieving session information for 51
 - running multiple instances of 7, 35
 - selecting skin for 58, 60, 64
 - setting options for 133, 141
 - starting 8
 - viewing available locales for 86
 - viewing changes to 49
- Information Console custom pages. *See* web pages
- Information Console Security Extension 238
- Information Console technology 4
- Information Delivery API 4, 8
- information objects 118, 124, 160, 225
- init method 231
- input 42, 50
- INSTALL_MODE parameter 78
- installing Information Console 5, 6, 78
- instanceid parameter 187
- Interactive Viewer 17, 57, 94, 186
- Interactive Viewer servlet 186
- InteractiveViewing subfeature 17, 85
- Intermediate functionality level 14, 15, 83
- internationalization. *See* locales
- internationalization tag library 194, 195, 196
- internet applications. *See* web applications
- invokeSubmit parameter 128, 157
- IP addresses 239
- iportal context root 31
- iportal directory 32, 33
- iPortalID parameter 101

- iPortalLogin action 107
- iPortalRepository class 52
- iPortalSecurityAdapter class 243
- IPSE applications 238
- isAlwaysGetFolderList method 232
- isEanalysisOptionEnabled method 232
- isEnterprise method 52, 245
- iServer
 - balancing workload on 6, 7, 39
 - connecting to 80, 89, 210, 233, 239
 - getting output formats for 202
 - getting security credentials for 244, 245
 - installing Information Console with 5, 78
 - logging in to 239
 - running Information Console and 4, 14
 - running queries on 125, 159
 - sending requests over 8, 11, 32
- iServer System 5, 30
- isFileTypeAllowed method 247, 249
- isHomeFolderSet method 232
- isIE method 229
- isLastRow variable 210
- isNS4 method 229
- isNS6 method 229
- isnull parameter 128
- isShowFilters method 232
- isViewInNewBrowserWindow method 232
- iterator tag 195, 209
- IteratorTag class 209
- iv action 107
- iv_config.xml 74
- IVServlet. *See* Interactive Viewer servlet

J

- Jakarta Struts action mapping 98, 102
- Jakarta Struts code 55
- Jakarta Struts Framework 48, 51
- Jakarta Struts templates 52, 53, 55
- Java classes 48
- Java Report Engine Manager 33
- Java Servlet API 48
 - See also* servlets
- JavaBean methods 224
- JavaBeans 52, 53, 66, 224, 232
- JavaBeans class reference 224
- JavaBeans package reference 224

- Javadoc 224
- JavaScript code 30, 48
- JavaScript files
 - accessing 52, 55
 - changing 50
 - creating online help and 253, 255
 - developing web applications and 172
 - referencing images and 69
- JavaScript reference 172
- JavaServer Pages. *See* JSPs
- JDK. *See* Java Development Kit
- job action forms 225, 227
- job classes 227
- JobActionForm class 225, 227
- jobID parameter
 - delete job page 119
 - delete status page 120
 - request detail page 121
 - request drop page 123
 - requests index page 134
- jobName parameter
 - delete job page 119
 - delete status page 120
 - execute query page 125
 - execute report page 128
 - request drop page 123
 - schedule page 153
 - submit job page 157
- joboperationstatus.jsp 119
- JobPriority subfeature 17, 85
- jobs
 - See also* requests
 - canceling 123
 - deleting 119, 123
 - displaying 117, 134, 150
 - executing 10
 - filtering 134
 - getting information about 121, 227
 - listing pending 146, 153
 - removing notifications for 119
 - running immediately 125, 128, 157
 - scheduling 100, 116, 152, 158
 - sending notifications for 17, 134, 157, 180
 - setting priorities for 17, 126, 128, 158, 181, 233
 - setting properties for 227

- jobs (*continued*)
 - submitting 10, 84, 133, 155, 225, 227
 - viewing parameters for 145
- Jobs attribute (features) 16, 84
- Jobs link 19
- jobs pages 134
- jobState parameter 119, 120, 123
- JSP code 55, 194
- JSP custom tag reference 194, 195, 196
- JSP engine 5, 8, 34
- JSP extensions 48
- JSP file names 57
- JSPs
 - accessing 32
 - changing templates and 55
 - compiling 30
 - customizing 32, 52, 54, 55
 - displaying 50
 - generating web pages and 10–11, 30, 56
 - getting input from 50
 - getting session information and 52
 - implementing URIs and 31
 - implementing URLs and 30
 - linking styles in 65, 66
 - locating specific 50
 - mapping actions to 98, 104
 - naming 98
 - referencing images in 69
 - selecting templates for 32, 53
 - updating changes to 49
- JUL_LOG_CONSOLE_LEVEL parameter 78
- JUL_LOG_FILE_COUNT parameter 78
- JUL_LOG_FILE_LEVEL parameter 78
- JUL_LOG_FILE_SIZE_KB parameter 78

K

- keepROIIfFailed parameter 157
- keepROIIfSucceeded parameter 157
- key attribute 211

L

- label keys 18, 84
- Labelkey tag 84
- labels 42, 84, 87
- landing page 48–49, 53
- landing.jsp 32, 49

- language codes 86, 87
- languages 74
 - See also* locales
- large icons 19, 84
- Largelcon tag 19
- LaunchHelp method 259
- launchiv parameter 187
- layer.js 172
- leading spaces 199
- Level tag 18, 83
- libraries 33, 52, 194
- license page 135
- license.jsp 135
- limit parameter 128, 181
- limitNumber parameter 128, 181
- Link tag 19, 66, 85, 226
- LinkBean class 226
- linking style definitions 65, 66
- linking to files 57
- linking to folders 131
- linking to web pages 56, 88
- links 8, 49, 82, 131, 256
 - See also* hyperlinks
- Linux systems 5, 31, 44, 47
- list package 224
- list pages 99, 113, 136
- lists 77, 78, 136, 137, 209, 224
- llformats variable 203
- load balancing 6, 7, 39
- locale codes 86, 87
- locale IDs 86
- locale names 86
- locale parameter 102, 184, 187, 190
- Locale tag 86
- localemap.xml 74, 86, 87
- locales
 - accessing repository for 30
 - accessing resources for 18
 - adding 86
 - bundling resources for 196, 211
 - changing 143
 - configuring 86
 - creating error messages for 87–88
 - formatting data for 201
 - generating reports for 184, 190
 - setting default 39, 76, 86, 232

- setting global styles for 57
- setting Reportlet 207
- specifying current 102, 197
- specifying preferred 197
- translating reporting applications for 42
- localhost value 11
- localizing messages 42, 87–88
- log file numbers 78
- log files 32, 77, 78
- LOG_FILE_LOCATION parameter 78
- logging in to
 - Information Console 12, 139, 240
 - iServer 239
- logging levels 78
- login action 9, 103, 107
- login banner page 99, 139
- login credentials 114, 210, 243, 244
- login forms 229
- login information 76, 229, 240
 - See also* login credentials
- login page 12, 53, 99, 139
- login requests 239
- login tag 196, 210
- login tag library 195, 196
- LOGIN_TIMEOUT parameter 78
- login.do 113
- login.jsp 139
- LoginForm class 229
- loginPostBack parameter 139
- logins
 - customizing 238
 - forcing 101
 - getting run as user information for 244
 - getting user information for 229, 244, 245
 - redirecting 9, 139
- LoginTag class 210
- logo.gif 68
- logos 68, 261–264
- logout action 103, 107
- logout page 100, 140
- logout.do 113

M

- machine names 11
- magnification levels. *See* scalingFactor parameter

- magnifying glass icon 69
- Management Console 5, 16, 176
- MAX_BACKUP_ERROR_LOGS
 - parameter 78
- MAX_CONNECTIONS_PER_SERVER
 - parameter 89
- MAX_LIST_SIZE parameter 78
- MDS_ENABLED parameter 7, 39
- MDS_REFRESH_FREQUENCY_SECONDS
 - parameter 39
- MDS. *See* Message Distribution service
- MEASURE_ parameter 91
- measure_ parameter 128
- memory 77, 89
- menu items 22, 23, 24
- menus 16, 69, 85, 133
- Message Distribution Service 6, 39, 40, 90, 147
- message keys 42
- message tag 196, 211
- messages 42, 45, 132
 - See also* error messages
- messages.properties 42
- MessageTag class 211
- method calls 232
- methods 224
- mobile
 - accessing 84
- Mobile attribute (features) 16, 84
- MOBILE_APP_DOWNLOAD parameter 76
- mode parameter
 - ping page 148
 - save as page 151
 - view default page 165
 - view frame set page 167
 - ViewPage servlet 191
- Model-View-Controller architecture 30
- multi-byte characters 14
- My dashboard page 140
- My Documents folder 133
- My Documents link 133
- My Documents page 57
- My Folder icon 68
- My Folder link 131

N

name parameter

- DownloadFile servlet 178
 - DownloadSearchResult servlet 178
 - file or folder detail page 121
 - file or folder drop page 123
 - GetDynamicData servlet 182
 - GetReportData servlet 184
 - GetStaticData servlet 185
 - print page 149
 - privileges page 150
 - search report page 163
 - view cube page 160
 - view default page 165
 - view frame set page 167
 - view navigation page 168
 - view TOC page 170
 - ViewPage servlet 191
- names 57
- See also* file names; user names
- naming
- experience levels 93
 - functionality levels 16, 83
 - JSPs 98
 - output files 128, 145, 158, 181
 - query output 125
 - skins 62
- naming restrictions 30, 98, 155, 176, 194
- NAT routers 239
- native2ascii utility 87
- navigation bars 165, 166
- navigation modes 151, 165, 167
- navigation toolbars 168
- Network Address Translation (NAT) 239
- networks 4, 6, 79, 238, 239
- new request index page 133
- newKey parameter 143
- newLocale parameter 143
- newrequest directory 10
- newrequestpag.jsp 56
- newTimeZone parameter 143
- notification groups 180
- notification page 100, 141
- notificationAttachment parameter 157
- notifications
- deleting 119

- generating 142
 - iterating through 210
 - sending 17, 134, 157, 180
 - setting options for 141
 - specifying user names for 129, 134
 - storing 227
- notificationSupported parameter 157
- notify parameter 157
- Novice experience level 20, 92
- null values 128
- NUMBER_OF_LEVELS tag 26, 93

O

- object types. *See* type parameter
- objectID parameter
- file or folder detail page 121
 - save as page 151
 - view navigation page 169
 - ViewEmbeddedObject servlet 188
 - ViewPage servlet 191
- objects 151, 187, 188, 194
- oldKey parameter 143
- onceDate parameter 153, 158
- onceTime parameter 153, 158
- on-demand paging. *See* immediate jobs;
- progressive viewing
- online help 252–276
- onlyLatest parameter 138
- open source frameworks 48
- opening
- Analytics Cube Viewer 160
 - help files 254
 - Interactive Viewer 17
 - login page 101
 - skin manager 59
 - web applications 35, 132
 - web browser windows 143, 235
- operation parameter 184, 191
- options 11, 233
- options action 107
- options directory 11
- Options functionality 133
- options index page 133
- options page 100, 141, 229
- options save action 107
- options update pages 32

- options.do 112, 113
- options.jsp 133
- output
 - archiving query 125
 - generating 125, 143, 183, 190
 - overwriting 125, 128, 144, 157, 181
- output file names 145
- output files
 - See also* report files
 - creating 125
 - deleting 126, 127, 156, 180
 - limiting number of 125, 128, 181
 - naming 128, 145, 158, 181
 - saving 129, 143, 150, 181
- output formats
 - executing queries and 126
 - generating reports and 156, 183
 - getting 202
 - saving reports and 150
 - viewing reports and 190
- output page 100, 143
- output types 205
- outputFolderType parameter 128, 145
- outputFormat parameter 158
- __outputName parameter 158
- outputName parameter
 - DownloadSearchResult servlet 179
 - execute query page 125
 - execute report page 128
 - ExecuteReport servlet 181
 - output page 145
 - submit job page 158
- oversize pages 192
- overwrite parameter 128, 181

P

- packages 224
- page breaks 76
- page counts 203
- page engine 8
- page names 98
- page navigation modes 151, 165, 167
- page not found messages 132, 145
- page not found page 100, 145
- page parameter
 - Interactive Viewer servlet 187
 - print page 149
 - save as page 151
 - view default page 165
 - view frame set page 167
 - view navigation page 169
 - ViewPage servlet 191
- page ranges 149, 151, 187, 191
- page styles 187
- pageHeight parameter 191
- pagenotfound.jsp 145
- pageWidth parameter 191
- parameter definitions 38
- parameter.css 66
- parameters
 - accessing data cubes and 225
 - adding to URIs 12, 13, 101
 - changing 38
 - configuring Information Console and 74
 - connecting to Encyclopedia and 30, 89
 - customizing reporting applications and 38
 - displaying 145, 158
 - getting specified 209
 - loading JSPs and 30
 - loading web pages and 10, 12
 - referencing report 155
 - returning session information and 51
 - running Analytics Cube Viewer and 90, 91
 - running reports and 94, 127, 130
 - setting up report viewers and 94
- parameters list 145
- parameters page 65, 100, 145
- partitionName parameter 148
- partitions 148
- password parameter 102
- passwords
 - adding to URIs 102
 - getting 230, 244
 - setting 233
 - updating 142, 240
- paths
 - context roots 51
 - dashboards 89
 - home folders 131
 - icons 19
 - image files 69
 - log files 78
 - temporary files 79, 90

- paths (*continued*)
 - title pages 262
- payloadSize parameter 148
- PDF files 149, 150, 184, 191
- PDF format 156, 184, 190
- PDFQuality parameter 184, 191
- pending jobs 146, 153
- pending page 100, 146
- pendingjob.jsp 146
- performance 77
- ping action 107, 227
- ping modes 148
- ping page 100, 146
- ping.do 113
- pop-up menus 23, 24
- popupmenu.js 172
- ports 8, 11
- postback parameter 125, 158
- preferences 101
- prefix attribute 194
- PRELOAD_ENGINE_LIST parameter 79
- presentation models 30
- previewing application pages 49
- print page 100, 149
- print.jsp 149
- printing 149
- prioritizing jobs 17, 126, 128, 181, 233
 - See also* priority parameter
- priority parameter
 - execute query page 126
 - execute report page 128
 - ExecuteReport servlet 181
 - submit job page 158
- priority settings. *See* priorityValue parameter
- priorityValue parameter
 - execute query page 126
 - execute report page 128
 - ExecuteReport servlet 181
 - submit job page 158
- private cache 75
- private directory 32
- private networks 239
- privileges 5, 149, 155, 210, 226
- privileges page 100, 149
- Process Management Daemon 140
- process redirect page 9
- processed action status page 123

- processID parameter 148
- processRedirect.jsp 9
- ProfileBean class 228
- profiles 40, 80, 90, 228, 233
- programmers 4
- progressive parameter 126, 129, 158
- progressive viewing 79, 126, 129, 158
- PROGRESSIVE_REFRESH parameter 79
- PROGRESSIVE_VIEWING_ENABLED
 - parameter 79
- prompts 42
- properties 233
- properties files 42
- protecting data 238
- proxy servers 7, 8, 9, 79, 239
- PROXY_BASEURL parameter 79
- public skins 60
- put tag 68

Q

- queries
 - archiving output 125
 - copying 159
 - creating 118, 225
 - formatting output for 126
 - generating output for 125
 - overwriting output for 125
 - running 124, 125, 159
 - submitting jobs for 155
- query action forms 225
- query classes 225
- query create action 108
- query definitions 225
- query execute action 108
- query pages 53, 54, 118, 125
- query submit action 108
- query.js 172
- QueryActionForm class 225, 227
- queryexecute.do 112
- querytemplate.jsp 53, 54

R

- range parameter 149, 151, 191
- recurringDay parameter
 - execute report page 129
 - ExecuteReport servlet 181

- schedule page 153
 - submit job page 158
- recurringTime parameter 153, 158
- redirect attribute 9
- redirect parameter
 - delete job page 119
 - delete status page 120
 - ExecuteReport servlet 181
 - file or folder drop page 123
 - options page 143
 - request drop page 123
 - submit job page 158
- redirection 9, 76, 139, 157
- redirects 257
- referencing
 - files 57
 - images 69
 - report parameters 155
- refresh intervals 40, 90
- relative hyperlinks 57
- renaming
 - files 57, 69
- report document files 121, 138, 156, 186, 234
- report emitters 75
- report executable files 138, 234
 - See also* executable files
- report execution requests. *See* jobs; requests
- report files
 - See also* specific type
 - accessing 10, 84, 137, 226
 - archiving 127, 156, 180
 - assigning privileges to 149
 - copying 200
 - deleting 17, 119, 122
 - displaying 138
 - downloading 177
 - filtering 137
 - getting information about 121, 226
 - linking to 57
 - overwriting 128, 144, 157, 181
 - preserving 157
 - saving 129, 143
 - searching 154
 - sharing 17
- report libraries 33, 194
- report object executable files 145
 - See also* executable files
- report object instance files 156
 - See also* report document files
- __report parameter 187
- report parameters 145, 155, 158
- report server. *See* iServer
- report viewers 95, 98, 161, 186
 - See also* specific viewer
- report.js 173
- reporting applications. *See* applications
- reporting system. *See* iServer System
- Reportlet format 190
- Reportlet tag library 195, 196
- reportletMaxHeight parameter 151, 185, 191
- Reportlets
 - displaying 204
 - getting size 204
 - redirecting 207
 - retrieving data for 183, 206
 - scaling 206
 - setting locale for 207
 - setting size 151, 191, 205, 207
- reports
 - adding table of contents for 170, 208
 - deploying 5
 - displaying 10, 11, 32, 79, 94, 95, 186
 - downloading 150, 177
 - filtering 137
 - getting page count for 203
 - paging through 151, 165, 167
 - printing 149
 - refreshing 79
 - retrieving data for 182, 185, 189
 - retrieving specific pages 151, 191
 - running 77, 126, 153, 179
 - saving 129, 143, 150, 181
 - scaling 152, 165, 167, 169
 - searching 84, 162, 212
 - submitting requests for 8, 126, 179
 - viewing specific pages 188
- repositories
 - See also* Encyclopedia volumes
 - accessing items in 10, 84, 137
 - displaying information about 56
 - downloading from 187
 - returning type 52
- REPOSITORY_CACHE_TIMEOUT_SEC
 - parameter 90

- repositoryType parameter 187
- request detail page 121
- request drop page 123
- Request page 143
- request search page 100, 162
- request variable 51
- requestFilters parameter 143
- requests
 - See also* jobs
 - dropping 123
 - failing 142
 - limiting number of items returned 78
 - loading web pages and 10, 12
 - running multiple applications and 7
 - sending 6, 11, 30, 32, 35
 - specifying action paths for 57
 - submitting 8, 116, 126, 153, 179, 239
 - testing Encyclopedia and 148
 - timing out 76
- requests index page 133
- requestsearch.js 173
- requestsicon.gif 19
- resetFilter parameter 134, 138
- resize.js 173
- resource bundles 196, 211
- resource files 18, 42
- resources 10, 18, 30, 55, 80
- resources.jar 18, 87
- restarting Apache Tomcat service 37
- restarting servlet engine 28
- reverse proxies 9
- RGB color values 61, 66
- rgb method 66
- rich text formats. *See* RTF formats
- .roi files. *See* report object instance files
- Role tag 16, 83
- roles
 - creating 233
 - defining functionality levels and 16
 - setting privileges for 155, 180
- RosFileID parameter 131
- RosFileName parameter 131
- .rox files. *See* report object executable files
- RTF files 150
- RTF formats 156, 184, 190
- RTFFullyEditable format 184, 190

- rtl parameter 187
- run requests. *See* jobs; requests
- running
 - Java servlets 176
 - jobs 10
 - queries 124, 125, 159
 - reports 77, 126, 153, 179
- running page 100, 150
- runningjob.jsp 150

S

- save as page 100, 150
- saveas.js 173
- saveInVolume parameter 181
- saveOutput parameter 129
- saving
 - image files 228
 - output files 129, 143, 150, 181
- scalingFactor parameter
 - GetDynamicData servlet 182
 - GetReportData servlet 185
 - save as page 152
 - view default page 165
 - view frame set page 167
 - view navigation page 169
 - ViewEmbeddedObject servlet 188
 - ViewPage servlet 191
- schedule page 100, 152
- schedule properties 152
- scheduled job page 100, 153
- scheduledjob.jsp 153
- schedulePeriod parameter 159
- scheduleType parameter 153, 159
- scheduling jobs 100, 116, 152, 158
- Search attribute (features) 16, 84
- search conditions 162, 163
 - See also* searchCriteria parameter
- search dialogs. *See* search forms
- search expressions 154, 164
- search file page. *See* file list page
- search folders page 100, 154
- search forms 162, 164, 166, 168
- search frame page 100, 162
- Search link 15, 19
- Search page 226
- search report page 100, 162

- search results
 - displaying 57, 131, 162
 - downloading 178
 - formatting 178, 213
 - returning 212
- search toolbar page 100, 164
- search toolbars 164
- search.js 173, 255
- searchCriteria parameter 166, 168, 169
- searchfiles action 109, 226
- searchfiles page. *See* file list page
- searchfiles.do 19, 113
- SearchFilesActionForm class 226
- searchFilter parameter 154
- searching
 - folders 84, 154
 - Reportlets 206
 - reports 84, 162, 212
 - specific components 152, 191, 198, 199
 - specific pages 152, 168, 169
- searchList parameter 152, 163, 191
- searchReport tag 196, 212
- searchReport URIs 163
- SearchReportTag class 213
- SearchTag element 163
- SearchTag property 163
- searchtoolbar.js 173
- security 4, 75, 238
- security adapter class 239, 241
- security adapters 239, 240–243
- security manager 243
- security roles. *See* roles
- SECURITY_ADAPTER_CLASS
 - parameter 79
- selectchannels action 108
- selectchannels page. *See* channels list page
- selectchannels.do 19, 113
- selectjobnotices action 109, 227
- selectjobnotices page. *See* channel contents list page
- SelectJobNoticesActionForm class 227
- selectjobs action 109, 227
- selectjobs page. *See* requests index page
- selectjobs.do 19, 112
- selectjobs.jsp 133
- selectUsers tag 214
- SelectUsersTag class 214
- SelfNotificationWithAttachment
 - subfeature 17, 85
- sending notifications 17, 134, 157, 180
- sending requests 11, 30, 32, 35
- server parameter 148
- server profiles 40, 80, 90
- server URLs 233
- servers
 - See also* iServer
 - accessing 52, 238
 - balancing workload among 7, 39
 - configuring context root for 35
 - deploying Information Console over 4, 5, 6, 238
 - dropping connections to 76
 - extending functionality of 176
 - installing Information Console on 5
 - maintaining session states for 7
 - optimizing performance for 77
 - restarting 36, 37
 - retrieving session information for 51
 - running multiple applications and 7, 35
 - sending requests over 8, 10
 - setting default 40
 - setting up firewalls and 8, 9, 238
 - updating user settings for 141
- serverURL parameter
 - execute report page 129
 - ExecuteReport servlet 181
 - Interactive Viewer servlet 187
 - submit job page 159
 - URIs 102
- services 146, 148
- servlet engine 5, 8, 28, 35, 50
- servlet names 176
- servlets 10, 30, 176–192
- servlets reference 177
- session attributes 137
- session information 51, 140
- session state 7
- session variables 138
- SESSION_DEFAULT_PARAMETER_
 - VALUE_ID parameter 79
- sessions 7, 76, 78, 79
- sessionTimeout parameter 79
- setAcLocale method 232
- setAlwaysGetFolderList method 232

- setAuthid method 232
- setCurrentfolder method 232
- setDefaultAnalyticsExpLevel method 232
- setDefaultServerURL method 233
- setDefaultVolume method 233
- setFeatureOptions method 233
- setFilter method 233
- setHomefolder method 233
- setMaxJobPriority method 233
- setOnlylatest method 233
- setPassword method 233
- setProfile method 233
- setProperty method 233
- setQueryDefinition method 225
- setRequest method 229
- setRoleNames method 233
- setServerurl method 233
- setShowdocuments method 234
- setShowexecutables method 234
- setShowFilters method 234
- setShowfolders method 234
- setSideBarFeatures method 234
- setSidebarSelected method 234
- setSkinConfig method 234
- setSkinName method 234
- setSystemname method 234
- setTimezone method 234
- setUserAgent method 234
- setUserid method 234
- setView method 235
- setViewInNewBrowserWindow method 235
- setVolume method 235
- ShareFile subfeature 17, 85
- showDocument parameter 138
- showExecutables parameter 138
- showFolders parameter 139
- showSearch parameter 166, 168
- showTOC parameter 166, 168
- side bars. *See* side menu
- side menu 16, 69, 85, 133
- simpletemplate.jsp 53, 54
- skin classes 227
- skin descriptions 63
- skin editor 228
- skin manager 47, 48, 59–60, 62, 228
- skin names 93, 234
- skin.config file 228

- SkinBean class 228
- skincustomization.js 173
- skinedit action 109
- SkinEditorActionForm class 228
- skinerror action 103
- SkinManagerActionForm class 228
- SkinManagerInfoBean class 228
- skins
 - accessing templates for 53
 - adding background images to 68
 - applying style definitions to 65, 66
 - changing application pages and 49, 57
 - changing images and 68
 - changing side menu for 85
 - cloning 60, 61
 - creating 59, 60, 61, 234
 - customizing 57, 59, 60, 62, 84, 228
 - deleting 60
 - editing 228
 - naming 62
 - previewing 60, 61
 - selecting 58, 60, 64
 - specifying default 60, 64
- skinstyles.css 65
- small icons 19, 84
- SmallIcon tag 19
- SmartSearch feature 131
- SOAP messages 45
- source code 52
- space character 13
- special characters 12
- splitOversizePages parameter 192
- spreadsheet reports 76, 80, 179
 - See also* Excel spreadsheets
- spreadsheet server. *See* iServer
- SQL statements. *See* queries
- Standard experience level 20, 92
- Standard Viewer 186
- start folders 210
- startFolder variable 211
- starting Information Console 8
- startingPoint parameter 164
- startup messages 132
- startUpMessage parameter 132
- static data 185, 187
- streamName parameter 185, 188
- string substitution 19

- string tag 195, 215
- stringList tag 195, 215
- StringListTag class 215
- strings 45, 101, 129, 199, 215
- StringTag class 215
- Struts action mapping 98, 102
 - See also* Jakarta Struts Framework
- strutscommon.js 173
- struts-config.xml 30, 50, 103
- style definition files 65
- style definitions 65, 66
- style sheets
 - accessing 52, 55
 - changing styles in 65
 - customizing web pages and 48, 68
 - embedding 151, 188
 - linking to JSPs 65, 66
 - specifying color settings in 66
 - updating changes to 50
 - viewing changes to 67
- STYLE tag 65, 66, 228
- styles 65–68
 - See also* style sheets
- styles directory 67
- subdirectories 32
- SubfeatureID tag 17, 85
- subfeatures 17, 85
- subfolders 154
- submit job page 100, 155
- submit page 100, 159
- submit.do 114
- submitjob action 109, 227
- submitjob page. *See* submit job page
- SubmitJobActionForm class 225, 227
- submitjobstatus.jsp 155
- submitquery action 103
- submitting jobs 10, 133, 155, 225, 227
- subpage parameter 132, 134
- SubscribeChannel subfeature 17, 85
- SubscribeChannelActionForm class 225
- succComp parameter 143
- succEmail parameter 143
- summary values 225
- SummaryBean class 225
- synchronizing data 17
- system names 234

T

- Tab class 216
- tab-separated formats. *See* TSV formats
- tab tag 195, 215
- tabbed dialogs
 - See also* tabs
 - defining tabs for 215, 218, 219, 221
 - setting orientation of 220
 - specifying content for 200, 219
- tabbed property sheets 133
 - See also* tabbed dialogs
- Tabbed skins 58
- TabBegin class 216
- tabBegin tag 195, 216
- TabEnd class 217
- tabEnd tag 195, 217
- table of contents
 - accessing help topics and 269, 271
 - displaying 166, 168, 169
 - generating 170, 208
- TABLE tag 55, 57
- tableList action 109
- tables. *See* HTML tables
- TabMiddle class 218
- tabMiddle tag 195, 218
- TabMiddleSelected class 218
- tabMiddleSelected tag 195, 218
- tabPanel tag 195, 219
- TabPanelTag class 219
- tabs
 - See also* tabbed dialogs
 - associating pages with 219
 - choosing skins and 58
 - defining adjacent pairs 221
 - defining labels and keys for 215
 - moving focus to 220
 - selecting 133
 - setting attributes of 218
 - setting order of 216, 217, 218
 - specifying as default 219
- TabSeparator class 221
- tabSeparator tag 195, 221
- tag libraries 52, 194
- tag library descriptor 194
- tag lines. *See* headline parameter
- tag names 194

- taglib directive 194
- tags
 - adding locales and 86
 - adding time zones and 86
 - changing company logos and 261, 263
 - changing help topics and 268
 - defining experience levels and 26, 86, 92, 93
 - defining features and 18, 84
 - defining functionality levels and 16, 83
 - defining subfeatures and 17, 85
 - encapsulating frequent tasks and 194, 195
- targetPage parameter 12, 139
- temp directory 90
- TEMP_FOLDER_LOCATION parameter 90
- template element 53
- template files 54
- template tags 53
- template.jsp 53, 54
- templates
 - accessing 53
 - building JSPs and 32, 54, 55
 - changing landing page and 49
 - creating web pages and 56
 - customizing applications and 52–56
 - selecting 53
- templates directory 32, 53
- temporary files 79, 90, 148
- temporary server profiles 82
- testing
 - applications 35
 - services 148
- text 42, 45, 53, 87, 268
- text editors 26
- text files 87
- text messages 42, 45, 132
- third-party applications 7
- time stamps 126, 129, 148
- time zone IDs 87
- time zones 39, 76, 86, 102, 143, 234
- time-out values 76, 78, 79
- timeToDelete parameter
 - execute query page 126
 - execute report page 129
 - ExecuteReport servlet 181
 - submit job page 159
- timezone parameter 102
- TimeZone tag 87
- TimeZones.xml 74, 86
- title pages 262
- TITLE_LANDING_PAGE parameter 42
- title.css 68
- title.js 255, 267
- titles 42, 267
- TLD files 194
- toc.js 255
- toctree.js 173
- toolbar buttons 21
- toolbar.css 68
- toolbars 164
- topics.js 255
- toString method 226, 232
- trailing spaces 199
- transient files 79, 90, 148
- TRANSIENT_STORE_MAX_SIZE_KB parameter 79
- TRANSIENT_STORE_PATH parameter 79
- TRANSIENT_STORE_TIMEOUT_SEC parameter 79
- translating reporting applications 42
- treebrowser action 109
- Treeview skins 58
- truncated strings 101
- trusted names 11
- TSV formats 178, 179
- TSV value 213
- type parameter
 - GetDynamicData servlet 182
 - GetReportData servlet 185
 - GetStaticData servlet 185
 - save as page 152
 - search report page 164
 - view navigation page 169
 - ViewPage servlet 192

U

- unauthorized users 238
- Uniform Resource Identifiers. *See* URIs
- UNIX systems 5, 31, 44, 47
- updating
 - data 17
 - passwords 142, 240
 - user options 141

- web pages 49
- upgrades 20, 60
- upload security adapter 249
- upload security adapter class 246, 247
- upload security adapters 246–249
- UPLOAD_FILE_TYPE_LIST parameter 79
- UPLOAD_SECURITY_MANAGER
 - parameter 80
- uploadimage action 110
- uploading binary files 176
- uploading image files 228
- uri attribute 194
- URIs
 - accessing reporting applications and 35
 - adding parameters to 12, 13, 101
 - creating 11, 12
 - displaying feature-specific pages and 19
 - embedding objects and 189, 190
 - encoding characters and 12, 14
 - implementing 31
 - loading servlets and 176
 - locating specific JSPs and 50
 - obtaining list values and 137
 - overview 11, 98
 - Process Management Daemon and 140
 - redirecting logins and 139
 - redirecting web pages and 9
 - referencing in tag libraries 194
 - returning diagnostic information and 146
 - running reports and 127, 130
 - searching and 163
 - submitting requests and 8, 11, 32
 - viewing reports and 161, 183
- URIs reference 111, 161
- URLs
 - activating security manager and 240
 - connecting to iServer System and 30, 80, 233
 - opening help files and 256, 257
 - redirecting Reportlets and 207
 - redirecting web pages and 9, 76
 - setting up firewalls and 8
 - specifying default 233
 - transmitting actions and 30, 51
- useQuoteDelimiter parameter 164
- user authentication. *See* authentication
- user classes 228
- user IDs 102, 187, 234
- user interfaces
 - building 4, 53, 57, 194
 - changing elements in 68–70
 - enabling features for 14, 16, 82
 - enabling subfeatures for 17, 85
 - hiding features in 21
 - submitting requests and 8
- user names 245
 - See also* userName parameter
- user parameter 140
- user profiles 228, 233
- user-agent header 229
- userAgent parameter
 - GetReportData servlet 185
 - save as page 152
 - view default page 166
 - view frame set page 168
 - view navigation page 169
 - ViewPage servlet 192
- UserAgentBean class 229
- UserAgentBean objects 234
- userID parameter 102
- userid parameter 187
- UserInfoBean class 52, 229, 229–235
- userName parameter
 - delete status page 120
 - options page 143
 - request detail page 121
 - requests index page 134
- users
 - accessing home page 131
 - displaying current settings for 130
 - displaying preferences for 101
 - getting authentication IDs for 52
 - getting home folders for 245
 - getting passwords for 230, 244
 - getting security credentials for 244
 - returning information about 214, 229
 - selecting experience levels 18, 20, 28, 91
 - selecting functionality levels 228
 - sending notifications to 129, 134, 141, 157, 181
 - setting default skins for 60, 64
 - setting experience levels for 20, 93, 232
 - setting features for 84, 234

users (*continued*)

- setting functionality levels for 14, 15, 16, 82, 83
 - setting passwords for 233
 - updating passwords for 142, 240
 - updating settings for 141
 - validating credentials for 114, 243
 - viewing subscribed channels for 136, 142, 224
- users parameter 129, 181
- users tag library 194
- UTF-8 encoding 14

V

values. *See* data

variables 42, 45, 138

verifyFile method 247, 249

version names. *See* versionName parameter

version parameter

- DownloadFile servlet 178
- file or folder detail page 121
- GetDynamicData servlet 182
- GetReportData servlet 185
- GetStaticData servlet 185
- save as page 152
- search report page 164
- view default page 166
- view frame set page 167
- view navigation page 169
- ViewPage servlet 192

versionName parameter

- execute query page 126
- execute report page 129
- ExecuteReport servlet 181
- output page 145
- submit job page 159

view constants 235

view cube page 101, 160

view default page 101, 165

view frame set page 101, 166

view navigation page 101, 168

View service 148

view TOC page 101, 169

VIEW_XLS_IN_REQUESTER parameter 80

viewcube action 104, 110, 225

viewcube page. *See* view cube page

viewcube.do 114

ViewEmbeddedObject servlet 187

viewer directory 32

viewer getsavedsearch action 110

viewer navigation toolbar 168

viewer page 101

viewer savesearch action 110

viewer servlet 186

viewer tag library 195, 196

viewer.js 173

viewers 95, 98, 161, 186

See also specific Actuate viewer

viewFormat parameter 126

viewframeset action 104

viewframeset page. *See* view frame set page

viewframeset.js 173

viewframesetfuncs.js 173

viewing

- banners 115, 139
- color settings 64
- completed jobs 117
- cube reports 160
- current jobs 150
- data 30, 56, 182
- embedded objects 187, 188
- error messages 124, 145
- failed jobs 134
- files and folders list 77
- folders 139, 234
- help topics 271, 273
- images 184, 191
- locales 86
- login page 12, 139
- PDF files 149
- pending jobs 153
- report executables 138, 234
- report parameters 145, 158
- Reportlets 204
- reports 10, 11, 32, 79, 94, 95, 186
- repository information 56
- search results 57, 131, 162
- specific report pages 188
- spreadsheet reports 76
- subscribed channels 116, 136, 142, 224
- table of contents 166, 168, 169
- web pages 50

- viewnav.js 173
- viewNewBrowser parameter 143
- viewpage action 104
- ViewPage servlet 188
- views 91, 235
- viewsoi action 104, 110
- volume icons 68
- volume parameter
 - execute report page 129
 - ExecuteReport servlet 181
 - Interactive Viewer servlet 187
 - submit job page 159
 - URIs 102
- volume_icon.gif 68
- VOLUME_PROFILE_LOCATION
 - parameter 90
- VolumeProfile parameter 102
- VolumeProfile.xml 74
- VolumeProfiles tag 80
- VolumeProfiles.xml 40, 80
- volumes. *See* Encyclopedia volumes

W

- wait parameter 129, 181
- wait values 76, 77, 129, 181
- waitforreportexecution action 111
- WAR files 5, 6
- web applications 8, 48, 238
 - See also* applications
- web archive files. *See* WAR files
- web browsers
 - changing style definitions for 66
 - changing title bar text for 267
 - changing web pages and 57
 - deploying Information Console and 5
 - detecting 229
 - displaying environment settings for 130
 - displaying Reportlets with 204
 - displaying reports in 165, 166, 188
 - encoding and 12, 14
 - issuing URIs and 101
 - loading web pages for 10, 12
 - maintaining session state for 7
 - opening new windows for 143, 235
 - preserving login information for 76
 - redirecting 9, 76, 139, 157

- scaling reports for 152, 165, 167, 169
- setting cache for 75
- specifying 152, 166, 168, 169, 234
- web pages
 - adding 10, 51, 52, 56, 88
 - associating with tag libraries 194
 - caching 49, 75, 90
 - creating banners for 115
 - customizing 48, 55, 56, 88
 - displaying 50
 - embedding objects in 185, 187, 188
 - embedding Reportlets in 204
 - formatting 57
 - generating 10–11, 30, 56
 - linking to 56, 88
 - loading 10, 12
 - localizing 196, 211
 - navigating through 56
 - submitting requests and 8
 - updating 49
 - viewing changes to 49
- web resources 10, 30, 55
- web servers 5
 - See also* servers
- web sites 5
- web.xml 74, 94
- WEB-INF directory 74, 194
- webreporting.css 68
- welcome text 49
- wildcards 154
- windows, closing 167
- Windows language pack 88
- Windows systems 5, 31, 43, 46
- working folders 116, 137
- workingFolder parameter 116
- workingFolderID parameter 118
- workingFolderName parameter 118
- wrcontextmenu.css 68
- write tag 66
- wwhelp directory 253
- wwhelp.html file 257

X

- XML files 49
- XML tag reference 194, 196
- XML tags 194

XMLCompressedDisplay format 184, 190

XMLData format 184

XMLDisplay format 170

Z

zoom levels. *See* scalingFactor parameter