Sun Java™ System

Content Delivery Server 5.1 Integration and Configuration Guide

Sun Microsystems, Inc.
www.sun.com

Part No.: 820-1944-10
June 2008

Submit comments about this document at: `http://www.sun.com/sunsurveys/dsc/dsc-feedback.jsp`

Please
Recycle

Adobe PostScript

Contents

Preface xvii

Part I Infrastructure Set Up

1. Deployment Configuration 1–1

1.1 Configuring the Monitoring Service 1–1

1.2 Configuring the Event Service 1–3

1.3 Configuring Synchronization 1–4

1.4 Configuring Marketing Campaigns 1–5

1.5 Setting Up Custom Reports 1–5

1.5.1 Enabling Custom Reports 1–6

1.5.2 Data Stored for Reports 1–7

1.6 Configuring Externally Hosted Content 1–9

1.7 Setting Up Automatic Publishing 1–9

1.8 Setting Up Custom Fields 1–10

1.8.1 Defining Your Fields 1–11

1.8.2 Assigning Labels and Hints 1–13

1.9 Defining Popularity 1–14

1.9.1 Default Definition 1–14

1.9.2 Customizing the Popularity Definition 1–15

1.9.3 Scheduling the Recalculation of Popularity 1–16
1.10 Enabling Cleanup of Expired Items 1–17
1.11 Configuring Device Client Web Services 1–17
1.12 Configuring the Fulfillment Service 1–18

2. Billing Integration 2–1
2.1 Billing Adapters Provided 2–2
2.2 Working with the Postpaid Service 2–2
2.2.1 Configure the Postpaid Service 2–2
2.2.2 Billing Event Parameters 2–5
2.3 Configuring External Content and Group IDs 2–8

3. Subscriber Integration 3–1
3.1 Subscriber Adapter Provided 3–1
3.2 Using a Subscriber Adapter 3–2
3.3 Working With LDAP 3–2
3.3.1 Creating the Mapping File for LDAP 3–2
3.3.1.1 Subscriber Data for Content Delivery Server 3–3
3.3.1.2 Sample Mapping File 3–6
3.3.2 Configuring Access for Sun Java System Application Server 3–8
3.3.3 Tuning LDAP 3–8
3.4 Subscriber Account Management 3–8
3.5 Automatic Provisioning of Subscribers 3–11

4. DRM Server Integration 4–1
4.1 Setting Up DRM Fusion Toolkit 4–1
4.2 Configuring Content Delivery Server to Use DRM Fusion Toolkit 4–2
4.2.1 Setting the Configuration Properties 4–2
4.2.2 Copying DRM Fusion Toolkit JAR Files 4–2
4.2.3 Enabling OMA DRM 1.0 4–3

5. Streaming Configuration and Integration 5–1

5.1 Integrating With a Streaming Server 5–1

5.2 Configuring Content Delivery Server for Streaming 5–2

6. Single Sign-On Support 6–1

6.1 Using the MSISDN Number Through the WAP Gateway 6–1

Part II Messaging Configuration

7. WAP Gateway Configuration 7–1

7.1 WAP Gateway Adapters Provided 7–1

7.1.1 Default WAP Gateway Adapter 7–2

7.1.2 Nokia Activ Server 2.0.1 7–2

7.1.3 Nokia Artus WAP Gateway 7–2

7.1.4 Openwave WAP Gateway 7–2

7.2 Using a WAP Gateway Adapter 7–3

8. Outgoing Push Messages 8–1

8.1 Push Sender Adapters 8–1

8.1.1 SMS Push for Messages 8–2

8.1.1.1 Setting Up Support for SMPP 8–2

8.1.1.2 Setting Up Support for SMS HTTP 8–4

8.1.1.3 Setting Up Support for CIMD2 8–5

8.1.2 WAP Push for Messages 8–5

8.1.3 SMTP Push for Messages 8–6

8.1.4 SMS Push for Binary Content 8–6

8.2 Using a Push Adapter 8–6

8.3 Configuring the SMS Push Message 8–7

8.4 Configuring Support for MMS 8–8

9. Mobile Originated Push Messages 9–1

9.1 Push Listener Adapter 9–1
9.1.1 Configuring the Subscriber Portal for MO Push 9–2
9.1.2 Configuring the Messaging Service for MO Push 9–3
9.1.2.1 Setting Up Support for SMPP or CIMD2 9–3
9.1.2.2 Setting Up Support for HTTP 9–5
9.2 Using a Push Adapter 9–6

10. Messaging Service and Confirm Service Configuration 10–1
10.1 Configure the Mail Service 10–1
10.2 Configure Storage of Response Messages 10–2
10.3 Handling Confirmation Messages 10–2

Part III Content Management

11. Search Setup 11–1
11.1 Configuring Default Result Fields 11–1
11.1.1 Catalog Manager Administration Console 11–2
11.1.2 Developer Portal 11–5
11.1.3 Vending Manager Administration Console 11–5
11.1.4 Subscriber Portal 11–7
11.2 Configuring the Search Engine 11–7
11.2.1 Configuring the Default Search Fields 11–8
11.2.2 Adding Custom Fields to the Search Index 11–9

12. Pricing Configuration 12–1
12.1 Set the Currency Symbol 12–1
12.2 Set a Grace Period 12–3

13. Digital Rights Management Configuration 13–1
13.1 Setting the DRM Methods Supported 13–1
13.1.1 Content Delivery Server DRM Agents 13–3

13.1.2 OMA DRM 1.0 Methods 13–5

13.2 Setting the Preferred Delivery Type 13–5

13.3 Setting the Preferred Action for Devices that Do Not Support OMA DRM 1.0 13–6

13.4 Setting the Event Handler 13–7

14. Submission Verifier Workflows 14–1

14.1 Content Validation Adapters 14–1

14.1.1 Adapters Provided 14–2

14.1.2 Writing an Adapter 14–3

14.1.3 Registering a Content Validation Adapter 14–3

14.2 Workflows Provided 14–3

14.2.1 Workflow for Java Applications 14–4

14.2.1.1 Default Workflow for Java Applications 14–4

14.2.1.2 Workflow for Signing Java Applications 14–5

14.2.2 Workflow for iAppli Applications 14–5

14.2.3 Workflow for Externally Hosted Copyrighted Content 14–6

14.2.4 Default Workflow 14–6

14.3 Creating a Workflow 14–7

14.3.1 Using the Add Capability Adapter 14–8

14.3.2 Using the External to Internal Adapter 14–9

14.3.3 Using the Process OMA DRM Message Adapter 14–9

14.3.4 Using the Set Edition Weight Adapter 14–10

14.4 Specifying Workflow Criteria 14–11

15. Previews and Watermarks 15–1

15.1 Configuring Previews 15–1

15.1.1 Setting the Default Caption 15–1

15.1.2 Identifying Audio Preview Files 15–2

15.2 Configuring Watermarking 15–2

15.2.1 Installing the Java Advanced Imaging Image I/O Tools 15–3
15.2.2 Watermarking When Content is Stocked 15–4
15.2.3 Watermarking When Content is Previewed 15–8

16. Content and Submission Formats 16–1
16.1 Configure iAppli Support 16–1
16.1.1 Add the DoJa Library to the Database 16–1
16.1.2 Submit the DoJa Library 16–2
16.2 Configure PAR File Support 16–3

17. Featured Content 17–1
17.1 Enabling Featured Content 17–1
17.2 Configuring Featured Content 17–3

Part IV User Interactions

18. Portal Configuration 18–1
18.1 Setting the Common Properties 18–2
18.2 Setting the Developer Portal Properties 18–3
18.3 Setting the Subscriber Portal Properties 18–4
18.4 Setting the Catalog Manager Properties 18–12
18.5 Setting the Vending Manager Properties 18–13

19. Device-Specific User Interface Framework 19–1
19.1 Overview of the Framework 19–1
19.1.1 Page Definitions 19–2
19.1.2 Style Sheets 19–10
19.1.3 Processes and Page Usage 19–11
19.1.3.1 Log In Process 19–12
19.1.3.2 View Content Process 19–13
19.1.3.3 Search for Content Process 19–14

19.1.3.4 Set Preferences Process 19–15
19.1.3.5 View Promotions Process 19–16
19.1.3.6 View the My Downloads List Process 19–16
19.1.3.7 View My Wish List Process 19–16
19.1.3.8 View the My Gifts List Process 19–17
19.1.3.9 Purchase or Download Process 19–17
19.2 Generating Pages for a Specific Device 19–20
19.3 Modifying Pages for All Devices 19–21
19.4 Adding a Custom Page 19–22

20. Notification Configuration 20–1
20.1 Configuring Developer Notifications 20–1
20.2 Configuring the Default for Subscriber Notifications 20–4

Index Index–1

Figures

FIGURE 15-1 Default Watermark 15–3
FIGURE 19-1 Log In Process 19–12
FIGURE 19-2 View Content Process 19–13
FIGURE 19-3 Search for Content Process 19–15
FIGURE 19-4 Set Preferences Process 19–15
FIGURE 19-5 Purchase Process 19–18

Tables

TABLE 1-1 Monitoring Service Properties 1–2

TABLE 1-2 Report Data 1–7

TABLE 1-3 Properties for Custom Fields 1–11

TABLE 2-1 Billing Event Parameters 2–5

TABLE 3-1 Configuration Properties 3–3

TABLE 3-2 LDAP Properties 3–3

TABLE 3-3 Subscriber Data 3–4

TABLE 3-4 Data for Sample LDAP File 3–6

TABLE 4-1 Properties for DRM Fusion Toolkit 4–2

TABLE 8-1 MMS Messaging Properties 8–9

TABLE 9-1 MO Push Message Parameters 9–2

TABLE 11-1 Field Names for the Catalog Manager 11–3

TABLE 11-2 Field Names for the Vending Manager 11–5

TABLE 12-1 Currency Properties 12–2

TABLE 12-2 Catalog Manager Currency Properties 12–2

TABLE 13-1 Files for Enabling and Disabling CDS DRM Agents 13–2

TABLE 13-2 CDS DRM Agents 13–3

TABLE 13-3 Delivery Type Settings 13–6

TABLE 13-4 Settings for Non-Compliant Devices 13–6

TABLE 14-1 Content Validation Adapters 14–2

TABLE 15-1 Properties for Defining a Watermark 15–4
TABLE 18-1 Properties in the `CommonConsole.properties` File 18–2
TABLE 18-2 Properties in the `DeveloperPortal.properties` File 18–3
TABLE 18-3 Properties in the `SubscriberPortal.properties` File 18–4
TABLE 18-4 Properties in the `AdminConsole.properties` File 18–12
TABLE 18-5 Properties in the `VSAdminConsole.properties` File 18–13
TABLE 19-1 XML Files for Subscriber Portal Pages 19–2
TABLE 19-2 Page Elements 19–4

Code Examples

CODE EXAMPLE 3-1 Sample Mapping File for LDAP Data 3-7

CODE EXAMPLE 3-2 Sample `subsubmgr.xml` File 3-9

CODE EXAMPLE 3-3 Sample `vsadminsubmgr.xml` File 3-10

CODE EXAMPLE 8-1 Sample `pushsenderfactory.xml` File 8-7

CODE EXAMPLE 9-1 Single Connection for MO Push Requests 9-4

CODE EXAMPLE 9-2 Sample `pushlistenerfactory.xml` File 9-6

CODE EXAMPLE 14-1 Sample Adapter Registration File 14-3

CODE EXAMPLE 14-2 Workflow for Externally Hosted Copyrighted Content 14-7

CODE EXAMPLE 14-3 Sample Workflow Step Using `AddCapabilityAdapter` 14-8

CODE EXAMPLE 14-4 Sample Workflow Step Using `ExternalToInternalAdapter` 14-9

CODE EXAMPLE 14-5 Sample Workflow Step and Criteria Using `ProcessOmaDrmMessageAdapter` 14-10

CODE EXAMPLE 14-6 Sample Workflow Step Using `SetEditionWeightAdapter` 14-11

CODE EXAMPLE 14-7 Criteria List for the Copyrighted External Content Workflow 14-11

CODE EXAMPLE 19-1 Sample Page With Form 19-9

CODE EXAMPLE 19-2 Sample Page With Links 19-10

CODE EXAMPLE 19-3 Terms and Conditions Page Definition 19-22

CODE EXAMPLE 19-4 Sample Handler 19-24

CODE EXAMPLE 20-1 Definition of Notification Templates 20-2

Preface

The *Sun Java™ System Content Delivery Server Integration and Configuration Guide* provides information about setting up Content Delivery Server to work within your infrastructure. Steps for configuring the features that you want to use are described, as are adapters that enable you to integrate Content Delivery Server with common systems and protocols that you are currently using.

Before You Read This Document

This guide is for system administrators or system integrators who are responsible for configuring Content Delivery Server and integrating it with their current infrastructure. It assumes some knowledge of networking, database, and wireless technologies, as well as experience with the Java programming language. You must have successfully deployed Content Delivery Server as described in the *Sun Java System Content Delivery Server Installation Guide* before using the information in this guide.

How This Document is Organized

This guide is divided into the following parts and chapters:

- Part I provides information on setting up your system.
  - Chapter 1 describes how to configure service components and some of the features of Content Delivery Server.
  - Chapter 2 describes billing adapters for integrating Content Delivery Server with your billing system.

- Chapter 3 describes subscriber adapters for integrating Content Delivery Server with your existing user data.
- Chapter 4 provides information on integrating the DRM application DRM Fusion Toolkit with Content Delivery Server.
- Chapter 5 provides information on setting up your system to support streamed content.
- Chapter 6 describes how Content Delivery Server supports single sign-on.

- Part II provides information on setting up messaging.
  - Chapter 7 describes WAP gateway adapters for configuring Content Delivery Server to support the WAP gateway that you use.
  - Chapter 8 describes push sender adapters for configuring Content Delivery Server to support the delivery method that you use.
  - Chapter 9 describes push listener adapters for configuring Content Delivery Server to accept mobile originated messages.
  - Chapter 10 provides information on configuring the Messaging Service and Mail Service.
- Part III provides information on setting up Content Delivery Server to manage content according to the needs of your enterprise.
  - Chapter 11 provides information on setting up the Content Delivery Server search services that create the search indexes for content and handle the browse and search queries.
  - Chapter 12 provides information on setting up the currency symbol and grace period that you want for pricing content.
  - Chapter 13 provides information on setting up the digital rights management (DRM) that you want to use.
  - Chapter 14 provides information on creating a customized submission verifier workflow to validate content submitted to Content Delivery Server.
  - Chapter 15 provides information on setting up your system to support preview files and watermarking.
  - Chapter 16 provides information on setting up your system to support different content types and submission formats.
  - Chapter 17 describes how to set up Content Delivery Server to support the concept of featured content in both the PC-based and device-based Subscriber Portal.
- Part IV provides information on setting up the portals for Content Delivery Server and configuring user interactions.
  - Chapter 18 describes some of the properties that are available for configuring the portals that are used to interact with Content Delivery Server.
  - Chapter 19 provides information for creating a version of the Subscriber Portal for a specific device.

- Chapter 20 provides information on setting up notifications to subscribers and content developers.

Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Typographic Conventions

Typeface	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your .login file. Use ls -a to list all files. % You have mail.
AaBbCc123	What you type, when contrasted with on-screen computer output	% **su** Password:
AaBbCc123	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this. To delete a file, type rm *filename*.

Note – Characters display differently depending on browser settings. If characters do not display correctly, change the character encoding in your browser to Unicode UTF-8.

Related Documentation

The following table lists the documentation for this product. The online documentation is available at:

http://docs.sun.com/app/docs/prod/cds

Application	Title	Part Number	Format	Location
Branding and localizing the portals	*Sun Java System Content Delivery Server Branding and Localization Guide*	820-1938-10	PDF HTML	`$CDS_HOME/Documentation/branding` and online.
Planning for your system	*Sun Java System Content Delivery Server Capacity Planning Guide*	820-1939-10	PDF HTML	`$CDS_HOME/Documentation/capacity` and online.
Working with the Developer Portal	*Sun Java System Content Delivery Server Content Developer Guide*	820-1940-10	PDF HTML	`$CDS_HOME/Documentation/devguide` and online.
Working with the APIs provided	*Sun Java System Content Delivery Server Customization Guide*	820-1941-10	PDF HTML	`$CDS_HOME/Documentation/customization` and online.
Trouble shooting	*Sun Java System Content Delivery Server Error Messages*	820-1942-10	PDF HTML	`$CDS_HOME/Documentation/errmsgs` and online.
Installing the system	*Sun Java System Content Delivery Server Installation Guide*	820-1943-10	PDF HTML	`$CDS_HOME/Documentation/install` and online.
Setting up and integrating with existing infrastructure	*Sun Java System Content Delivery Server Integration and Configuration Guide*	820-1944-10	PDF HTML	`$CDS_HOME/Documentation/integration` and online.
Migrating to the latest release	*Sun Java System Content Delivery Server Migration Guide*	820-1945-10	PDF HTML	`$CDS_HOME/Documentation/migration` and online.
Product reference information	*Sun Java System Content Delivery Server Reference Manual*	820-1946-10	PDF HTML	`$CDS_HOME/Documentation/refman` and online.
Monitoring and managing the system	*Sun Java System Content Delivery Server System Management Guide*	820-1947-10	PDF HTML	`$CDS_HOME/Documentation/system-mgmt` and online.

Documentation, Support, and Training

Sun Function	URL
Documentation	http://www.sun.com/documentation/
Support	http://www.sun.com/support/
Training	http://www.sun.com/training/

Third-Party Web Sites

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to http://docs.sun.com and clicking Send Comments.

Please include the title and part number of your document with your feedback:

Sun Java System Content Delivery Server Integration and Configuration Guide, part number 820-1944-10.

PART I Infrastructure Set Up

This part of the *Sun Java System Content Delivery Server Integration and Configuration Guide* provides information on configuring features of Content Delivery Server and integrating Content Delivery Server with your existing infrastructure.

This part contains the following chapters:

- Deployment Configuration
- Billing Integration
- Subscriber Integration
- DRM Server Integration
- Streaming Configuration and Integration
- Single Sign-On Support

CHAPTER 1

Deployment Configuration

After deploying Sun Java System Content Delivery Server, some configuration is needed to set up the deployment to work within your existing infrastructure. The configuration needed is dependent on the features that you want to use.

This chapter includes the following topics:

- Configuring the Monitoring Service
- Configuring the Event Service
- Configuring Synchronization
- Configuring Marketing Campaigns
- Setting Up Custom Reports
- Configuring Externally Hosted Content
- Setting Up Automatic Publishing
- Setting Up Custom Fields
- Defining Popularity
- Enabling Cleanup of Expired Items
- Configuring Device Client Web Services
- Configuring the Fulfillment Service

1.1 Configuring the Monitoring Service

To integrate the Application Monitoring Agent with your network monitoring system, you need to configure the Monitoring Service. Configure the Monitoring Service to issue only the alarms in which you are interested and disable all other alarms. The statuses and alarms issued by the Monitoring Agent are described in Section 1.1, "Application Monitoring Agent," in the *Sun Java System Content Delivery Server System Management Guide*.

To configure the Monitoring Service, edit the `CDSSnmp.properties` file in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory. The following table describes the properties.

TABLE 1-1 Monitoring Service Properties

Property	Description
`snmp.trap.disabled`	Indicates if all traps are disabled. Set to `true` to disable all traps and ignore the `snmp.trap.`*alarm*`.enabled` properties for the individual traps. Set to `false` to honor the settings for individual traps.
`snmp.trap.v1`	Indicates if Simple Network Management Protocol (SNMP) v1 or SNMP v2 traps are generated. Set this property to `true` to generate SNMP v1 traps. Set to `false` to generate SNMP v2 traps.
`snmp.agent.port`	The port number for the host on which the Monitoring Service is running.
`snmp.manager.address`	The IP address of the host on which the network management system is running.
`snmp.manager.port`	The port number for the host on which the network management system is running.
`snmp.oid.base`	The absolute object identifier (OID) value from the root node. Other Content Delivery Server OIDs are specified as a relative value from this base value. **Note -** Do not change this value.
Scalar Object Definitions	
status is the name of the component for which status is provided, for example, `eventServiceStatus`. See Section 1.1, "Application Monitoring Agent," in the *Sun Java System Content Delivery Server System Management Guide* for the valid statuses.	
`snmp.object.`*status*`.oid`	The MIB identifier for this object. **Note -** Do not change this value.
`snmp.object.`*status*`.handler`	The handler for this object. **Note -** Do not change this value.
`snmp.object.`*status*`.attr.pidfile`	The name and location of the file that contains the process ID for this object. **Note -** Do not change this value.
`snmp.object.`*status*`.attr.uri`	The path to the main servlet that this object monitors. **Note -** Do not change this value.

TABLE 1-1 Monitoring Service Properties *(Continued)*

Property	Description
Trap Definitions	
alarm is the name of the alarm that is issued, for example, `downloadFailure`. See Section 1.1, "Application Monitoring Agent," in the *Sun Java System Content Delivery Server System Management Guide* for the valid alarms.	
`snmp.trap.`*alarm*`.oid`	The MIB identifier for this object. **Note -** Do not change this value.
`snmp.trap.`*alarm*`.enabled`	Indicates if the alarm is issued. Set to `true` to permit the alarm to be issued. Set to `false` to prevent the alarm from being issued. The default is `true`. If `snmp.trap.disabled` is `true`, this setting is ignored.

1.2 Configuring the Event Service

The Event Service generates events based on the properties set in the `$CDS_HOME/deployment/`*deployment-name*`/conf/EventService.properties` file. If the consumer of the event is not running, the Java Message Service (JMS) maintains the events until the consumer of the event is started. Over time, messages might accumulate and result in an out-of-memory error.

The campaign event handler produces events that are consumed by the Notify Service. The billable event handler publishes messages to a topic for the Postpaid Service, which creates a durable topic listener client the first time it runs. The developer update event handler, developer submit event handler, and the OMA DRM 1.0 rights delivery event handler publish messages to a queue that is consumed by the Messaging Service.

If you do not plan to run the Notify Service, Postpaid Service, or the Messaging Service as part of your Content Delivery Server system, you can avoid the accumulation of messages by configuring the Event Service so the messages are not generated. To stop the generation of messages by a handler, edit the `EventService.properties` file and comment out the handler by adding a pound sign (#) to the beginning of the line for the property that defines the handler:

- When the Notify Service is not running, stop the generation of campaign notifications by commenting out the following statement as shown:

  ```
  #eventservice.handler=Campaign
  ```

- When the Postpaid Service is not running, stop the generation of billable events by commenting out the following statement as shown:

```
#eventservice.handler=Billing
```

- When the Messaging Service is not running, stop the generation of messages by commenting out the following statements as shown:

```
#eventservice.handler=DeveloperUpdate
#eventservice.handler=DeveloperSubmit
#eventservice.handler=OMARightsDelivery
```

1.3 Configuring Synchronization

The Catalog Manager and each Vending Manager has its own database. Information such as the status of content or the types of devices supported must be maintained across databases. Synchronization of the databases is managed by Content Delivery Server. When the Vending Manager receives a notification of change from the Catalog Manager, the Vending Manager attempts to synchronize with the Catalog Manager. If synchronization fails, the Vending Manager retries the operation.

To manage the number of times and how often the operation is attempted, set the following properties in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/RemoteVending.properties` file for the Vending Manager deployment:

- `vending.messaging.retries.no` - Number of times that the Vending Manager attempts to synchronize with the Catalog Manager if synchronization fails. After retrying for the number of times specified, the Vending Manager waits for the time interval specified for the `vending.messaging.processor.restart.interval` property before attempting another synchronization.
- `vending.messaging.processor.restart.interval` - Number of milliseconds the Vending Manager waits before trying again to synchronize with the Catalog Manager.

Note – A high number of retries or a short interval between each series of retries can slow response to subscriber requests and rapidly increase the size of log files.

Additional information about the synchronization between the Catalog Manager and Vending Managers is available in the “Catalog and Vending Managers” chapter of the *Sun Java System Content Delivery Server Reference Manual*.

1.4 Configuring Marketing Campaigns

Content Delivery Server provides a feature that enables you to send advertisements to selected subscribers. Messages can be sent using Short Message Service (SMS), Wireless Application protocol (WAP), Multimedia Messaging Service (MMS), or Simple Mail Transfer Protocol (SMTP).

Campaigns sent as email use the SMTP mail service defined for the Messaging Service. To change the mail service, see Section 10.1, "Configure the Mail Service" on page 10-1.

You can include a link to promotional items in the message. The address to which the link points is based on the value specified for the `sp.external.uri` property in your deployment configuration file.

If the value in the configuration file is not correct, edit the `CDS.properties` file found in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory and set the value for the `default.external.subscriberportal.uri` property to the correct address. If Content Delivery Server is running behind a firewall, specify an address that subscribers can access from outside the firewall.

1.5 Setting Up Custom Reports

The Vending Manager provides daily statistical reports that enable you to view and track application download information and usage statistics. To generate custom reports, you can configure Content Delivery Server to store transaction data in the `REPORT_DOWNLOAD` table in the Vending Manager database.

Note – To store transactions in the reports database, you must deploy and run the Postpaid Service.

1.5.1 Enabling Custom Reports

To configure support for customized reports, follow these steps:

1. **Edit the** `PostpaidService.properties` **file found in the** `$CDS_HOME/deployment/`*deployment-name*`/conf` **directory.**

   To enable the reporting handler, remove the pound sign (#) from the `postpaid.handler=ReportingHandler` statement.

   Also, set the following properties:

   - `postpaid.handler.ReportingHandler.events`. Specify one or more billing events for the reporting handler to process. The following events are valid:
     - `content_purchased`
     - `content_refunded`
     - `download_error`
     - `download_initiated`
     - `subscription_cancelled`
     - `subscription_purchased`
     - `usage_purchased`

     To specify more than one event, separate the events with a vertical bar (`|`).
   - `postpaid.handler.ReportingHandler.billingevent.process_free_downloads`. Set to `true` to process events for free content. Set to `false` if you do not want to process events for free content.
   - `postpaid.handler.ReportingHandler.billingevent.process_prepay_events`. Set to `true` to process events for prepaid content. Set to `false` if you do not want to process events for prepaid content.
   - `postpaid.handler.ReportingHandler.billingevent.identify_recurring_download_purchases`. Set to `true` to identify billing events for the per download billing model as Recurring. Set to `false` to identify the events as Purchased.
   - `postpaid.handler.ReportingHandler.recovery.starting.transaction`. Set to the time stamp that identifies the first transaction that you want to recover. The format of the time stamp is MM-dd-yyyy HH:mm:ss, which is defined in the `SimpleDateFormat` class. Past records are recovered when the REPORT_DOWNLOAD table is empty. The table is filled with data from the transaction that you specify to the most recent transaction.

2. **Edit the** `ReportService.properties` **file found in the** `$CDS_HOME/deployment/`*deployment-name*`/conf` **directory.**

   Set the following properties:

   - `reportdb.username`. The user name used to access the reports database. This is *prefix*`_vs_app`, where *prefix* is the value specified for the `Prefix` element under the `Vending` element in the database configuration file used to create the database. See the *Sun Java System Content Delivery Server Installation Guide* for information on this file.
   - `reportdb.password`. The password used to access the reports database. This is the value specified for the `Password` element under the `Vending` element in the database configuration file.
   - `reportdb.server.url`. The connection URL for the database. This value is specified as `jdbc:oracle:thin:@`*dbhost.domain.com*`:1521:`*sid*. Set *dbhost.domain.com* and *sid* as needed for your database server.

1.5.2 Data Stored for Reports

If you have enabled custom reports, the Postpaid Service stores a record of each purchase and refund transaction in the `REPORT_DOWNLOAD` table in the Vending Manager database. The information stored for each transaction is current at the time the transaction occurs. Information for stored transactions is changed only if a transaction recovery operation is performed and historical information is not available. For example, if a subscriber changes the device used, new transactions for that subscriber show the new device. Transactions that occurred with the old device continue to show the old device in the `MODEL` field, unless a recovery operation is performed. Recovered transactions show the new device.

The following table describes the data that is currently stored for each transaction. This table could change in future releases.

TABLE 1-2 Report Data

Name	Format	Description
`REPORT_DOWNLOAD_ID`	Number	Unique identifier for the transaction
`TRANSACTION_DATE`	Date	Time stamp of when the transaction occurred
`RESOURCE_INSTANCE_ID`	Number	Identifier for the edition of the content item involved in the transaction
`RESOURCE_CLASS_ID`	Number	Identifier for the content item involved in the transaction

TABLE 1-2 Report Data *(Continued)*

Name	Format	Description
CONTENT_NAME	String	Name of the content involved in the transaction
CONTENT_DESCRIPTION	String	Description of the content
CONTENT_TYPE	String	Type of content, for example, MIDlet
CONTENT_PROVIDER_KEY	String	Identifier that the content provider gave the content
CONTENT_PROVIDER_ID	Number	Identifier for the content provider
CONTENT_PROVIDER	String	Name of the content provider
CONTENT_COST	Float	Price assigned by the content provider
PRICE	Float	Price accepted by the subscriber
FULFILLMENT_REQUEST_ID	Number	Identifier for the fulfillment request
FULFILLMENT_STATUS	String	Status of the transaction, which is one of the following values: • Cancelled • Error • Initiated • Purchased • Recurring • Refunded • Requested • Trial
BILLING_MODEL	String	Billing model for the content
SUBSCRIBER_ID	Number	Identifier for the subscriber
MSISDN	String	MSISDN associated with the subscriber
FIRST_NAME	String	First name of the subscriber
LAST_NAME	String	Last name of the subscriber
MODEL	String	Name of the subscriber's device

1.6 Configuring Externally Hosted Content

If you support content that is hosted externally, the Event Service must have access to the Developer Portal. The value specified for the `dp.internal.uri` property in your deployment configuration file is used for this access. If this property is not set correctly, updates to externally hosted content are not fetched.

If the value in the configuration file is not correct, follow these steps to set the Developer Portal URL:

1. **Edit the `CDS.properties` file found in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory.**
2. **Set the value of the `default.internal.developerportal.uri` property to the internal address for the Developer Portal.**

1.7 Setting Up Automatic Publishing

Typically, the Catalog Manager administrator reviews all content that is submitted and determines which items to publish. If desired, you can configure Content Delivery Server to automatically publish content and bypass the review by the administrator. The following options are available:

- Automatically publish all content.

  For this option, set the `submission.content.auto_publish` property to `true` in the `$CDS_HOME/deployment/`*deployment-name*`/conf/DeveloperPortal.properties` file.

- Automatically publish content hosted externally when Content Delivery Server detects an update.

  For this option, set the `external.content.auto_publish` property to `true` in the `$CDS_HOME/deployment/`*deployment-name*`/conf/DeveloperPortal.properties` file.

- Automatically publish content as indicated by the rules set in the `$CDS_HOME/deployment/`*deployment-name*`/conf/AutoPublishRules.properties` file.

  For this option, define the rules for automatic publishing and create a submission verifier workflow that uses the Autopublish Content Validation Adapter. See the instructions in the `AutoPublishRules.properties` file for information on setting up the rules. See Chapter 14 for information on content validation adapters.

The following algorithm is used to determine whether submitted content is automatically published:

1. If the submission verifier workflow indicates that the content is to be published automatically, then `submission.content.auto_publish` and `external.content.auto_publish` properties are ignored and the content is automatically published.
2. If `submission.content.auto_publish` is `true`, then `external.content.auto_publish` is ignored and the content is automatically published.
3. If `external.content.auto_publish` is `true`, then the content is automatically published if the content is external content and the external content update was automatically detected by Content Delivery Server.
4. If `external.content.auto_publish` is `true` and the content is not external content, then the content is not automatically published.
5. Otherwise, the content is not automatically published.

1.8 Setting Up Custom Fields

Content Delivery Server provides a standard set of fields that provide information about each item of content, such as Name, Version, and Short Description. In addition to the standard fields, you can configure Content Delivery Server to manage custom fields.

Custom fields provide additional information about content that you require for your enterprise. Custom fields can also be used to support additional functionality. For example, Content Delivery Server uses custom fields to support the concept of featured content. See Chapter 17 for information on this feature.

You can define custom fields to apply to all content types or specify different fields for different content types. For example, you might want to create a field called Rating for the content type `midlet` to specify the rating of a game.

Note – Do not make any changes to the system reserved fields that are predefined by Content Delivery Server.

1.8.1 Defining Your Fields

To define a custom field, edit the `$CDS_HOME/deployment/`*deployment-name*`/conf/CustomField.properties` file and create the set of properties that are described in TABLE 1-3. For each property, *content-type* is the content type to which the field is associated, and *custom-key* is the string used to identify the custom field to which the property applies.

The content type specified must be a content type that is defined to Content Delivery Server. To associate the field with all content types, use `all`.

The custom key is an alphanumeric string that can also contain dash (-) or underscore (_) characters. A custom key can be used for more than one field only if the content type is different. If a key is used with the content type `all`, it must not be used with another content type.

TABLE 1-3 Properties for Custom Fields

Property	Description	Values
`emf.`*content-type*`.`*custom-key*`.scope`	Identifies whether the field applies to the content item or to the editions of the item. This property is required.	`item`, `edition`
`emf.`*content-type*`.`*custom-key*`.required`	Indicates if the field must have a non-null value. This property is required.	`true`, `false`
`emf.`*content-type*`.`*custom-key*`.datatype`	Specifies the data type that the field accepts. For `timestamp`, the valid format is `yyyy-MM-dd'T'HH:mm:ss.SSSZ`. This property is required.	`number`, `text`, `timestamp`, `boolean`
`emf.`*content-type*`.`*custom-key*`.editable`	Identifies the portals in which the field can be edited. Separate multiple values with a comma. If this property is omitted or no value is provided, the field is not editable in any portal. **Note -** If the `required` property is set to `true`, the value for the `editable` property must contain `dp`.	`dp`, `vm`, `cm`, empty string (“”)*
`emf.`*content-type*`.`*custom-key*`.viewable`	Identifies the portals in which the field can be viewed. If this property is omitted or no value is provided, the field is not viewable in any portal. **Note -** If a field is editable in a portal, it must also be set as viewable in that portal.	`dp`, `vm`, `cm`, `sp`, empty string (“”)*

TABLE 1-3 Properties for Custom Fields *(Continued)*

Property	Description	Values
`emf.`*content-type*`.`*custom-key*`.datasource`	Identifies the database in which this field is stored. For the Catalog Manager database, specify `cs`. For the Vending Manager database, specify `vs`. To specify both databases, separate the values with a comma. This property is required. **Note -** If the required property is set to true, the value must be `cs,vs` to store the value in both databases.	`cs`, `vs`
`emf.`*content-type*`.`*custom-key*`.weightage`	Indicates the order in which custom fields are shown. Fields are shown in ascending order based on the value assigned to this property. Values do not need to be consecutive. If two fields are given the same value, they are shown in random order within their position in the list. Fields that are not assigned a weightage are shown in random order before the fields that are assigned a weightage.	any integer

* `dp` = Developer Portal, `vm` = Vending Manager administration console, `cm` = Catalog Manager administration console, `sp` = Subscriber Portal, empty string ("") = no portal.

Note – If the definition of any custom field is invalid, an error message is shown when a user attempts to access content. The definition must be corrected to make content accessible.

To include custom fields in searches for content, the fields must be included in the search index. See Section 11.2.2, "Adding Custom Fields to the Search Index" on page 11-9 for instructions.

If you have remote Vending Managers, define custom fields for the Catalog Manager, then copy the `CustomField.properties` file to the Vending Managers. You can modify the properties of the custom fields for the Vending Managers, if desired, or define additional fields. Do not delete a field. If you do not want to show a field to the Vending Manager administrator or subscribers, remove the portal from the `viewable` and `editable` properties in the field definition. To hide a field from all portals, set the property to the empty string ("").

You do not need to restart the server after you make changes to `CustomField.properties`. If you change the properties after content has been submitted, existing content uses the new definitions, however, the changes could have the following consequences:

- Existing data in a custom field might become unavailable if the `datatype` property changes for a particular custom key.

- Custom fields that are added are empty. If a new field is declared `required`, errors might occur when processing existing content that does not have data for that field.
- Data in custom fields that are deleted becomes unavailable.

Content Delivery Server has no migration capability for custom fields. If data migration is required after you make changes to the custom field definitions, you must provide the process.

1.8.2 Assigning Labels and Hints

Labels and hints for custom fields are defined in a separate file for localization purposes. Labels identify the field and are shown in the user interface. Hints provide information about the field and are shown when a user hovers over the field with the mouse pointer. These strings must be set for each portal in the following property files, which are located in the `$CDS_HOME/deployment/`*deployment-name*`/localization` directory:

- `AdminConsoleMessages.properties`
- `DevPortalMessages.properties`
- `VendingManagerMessages.properties`
- `SubscriberPortalLocaleResources.properties`

The file names for localized versions of these files include the language code and possibly the country code. In the files for each portal and each supported language in which the field appears, set the following properties for each field that you defined in the `CustomField.properties` file:

- `emf.`*content-type*`.`*custom-key*`.label`. Set this property to the string that is shown as the label for the field.
- `emf.`*content-type*`.`*custom-key*`.hint`. Set this property to the string that provides information on the data to enter in the field. If this property does not exist for a field, a default hint based on the data type is used.

content-type is the content type to which the field is associated and *custom-key* is the string used to identify the field to which the attribute applies. The content type and custom key combination must match a field definition in the `CustomField.properties` file.

Tip – A default hint is provided for each data type. If you do not provide a hint for a field, the default hint for the data type associated with the field is used.

1.9 Defining Popularity

Knowing how popular an item of content is can be useful information. Content Delivery Server provides a function for calculating popularity based on the number of times content is accessed over time. If the default definition of popularity is not suitable for your needs, you can customize it. The change can be as simple as limiting the events that are included in the count of times accessed or as advanced as writing your own algorithm for calculating popularity.

The field that contains the popularity rating is available only for stocked content and is automatically updated by Content Delivery Server according to the schedule you set. This field can be shown in the Vending Manager administration console or in the Subscriber Portal by including it in the default search results or in a search query. See Section 11.1, "Configuring Default Result Fields" on page 11-1 for information on setting up the default search results.

Popularity ratings are specific to a Vending Manager. If multiple Vending Managers that do not share a database stock the same item of content, the popularity rating for that item is calculated independently by each Vending Manager.

1.9.1 Default Definition

By default, Content Delivery Server rates the popularity of content by dividing the number of system-generated events that are counted towards popularity by the time that has passed since the first event occurred. The result of this calculation is a floating point value. The following events are counted to determine popularity:

- `content_purchased`
- `subscription_purchased`
- `usage_purchased`
- `gift_download_initiated`
- `gift_purchased`
- `gift_subscription_purchased`
- `gift_usage_purchased`

Each time one of the events in the list is processed, the event count is increased by one. By default, events for free content are also counted.

Content Delivery Server uses custom fields that are defined in the `$CDS_HOME/deployment/`*deployment-name*`/conf/CustomField.properties` file for storing information related to popularity. Do not change field definitions for the fields with the following prefixes:

- `emf.all.first_access_time` - Stores the timestamp of the first time the content is accessed.
- `emf.all.popularity_hits` - Stores the number of events counted towards popularity for the content.
- `emf.all.popularity` - Stores the popularity rating for the content. This field is used to display the rating and is used to search or sort content by popularity.

1.9.2 Customizing the Popularity Definition

If the default definition of popularity does not meet your needs, you can create your own definition. You can continue to use the default algorithm and change only the events that are counted, or you can write your own algorithm.

The following properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/EventService.properties` file control which events are counted and whether free content is counted by the default algorithm:

- `eventservice.handler.Popularity.events`. Remove system-generated events that you do not want included in the popularity count and that cannot be excluded using one of the other properties. Do not add events to this property.
- `eventservice.handler.Popularity.count_download`. Set to `true` to include the `download_initiated` event in the popularity count. Set to `false` if you do not want the event included in the count.
- `eventservice.handler.Popularity.count_purchase`. Set to `true` to include the `content_purchased`, `subscription_purchased`, and `usage_purchased` events in the popularity count. Set to `false` if you do not want these events included in the count.
- `eventservice.handler.Popularity.count_gift`. Set to `true` to include the `gift_download_initiated`, `gift_purchased`, `gift_subscription_purchased`, and `gift_usage_purchased` events in the popularity count. Set to `false` if you do not want these events included in the count. If set to true, gift_download_initiated events are counted only if `eventservice.handler.Popularity.count_download` is also set to true.
- `eventservice.handler.Popularity.count_free_content`. Set to `true` to include the events for free content in the popularity count. Set to `false` if you do not want free content included in the count. If set to `true`, at least one other property must also be set to `true` for free content to be counted.

The default algorithm used by Content Delivery Server to set the popularity rating is defined in the `com.sun.content.server.vending.PopularityImpl` class. To define your own algorithm, implement the `com.sun.content.server.content.Popularity` interface that is part of the `cdsapi.jar` file. See the output of the Javadoc™ tool in the `$CDS_HOME/javadoc/cdsapi` directory for information on this interface.

When your implementation is ready, set the `vsadmin.popularity.impl` property in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/VSAdminConsole.properties` file to the fully qualified class name. Put the file that contains your algorithm in the `$CDS_HOME/deployment/`*`deployment-name`*`/lib/external` directory for Content Delivery Server to access.

If you want to change the way events are counted, for example, to add different weights to the events, you can write your own popularity handler to handle the events and set the count that is stored in the custom field `popularity_hits`. If you do write your own handler, set the `eventservice.handler.Popularity.classname` property in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/EventService.properties` file to the fully qualified class name.

1.9.3 Scheduling the Recalculation of Popularity

Recalculating the popularity of all content in a Vending Manager can decrease performance while the process runs. Instead of recalculating every time a relevant event is processed, configure Content Delivery Server to recalculate on a set schedule.

To set up the schedule, set the following properties in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/VSAdminConsole.properties` file:

- `vsadmin.popularity.reindex.interval`. Set this property to the number of minutes between recalculations.
- `vsadmin.popularity.reindex.start`. Set this property to the number of minutes past midnight for the base time from which the interval is calculated. For example, if this property is set to 30 and `vsadmin.popularity.reindex.interval` is set to 90, popularity ratings are recalculated every 90 minutes beginning at 12:30 a.m.

- `vsadmin.popularity.reindex.enable`. Set this property to `true` to recalculate popularity ratings on the schedule specified by the `vsadmin.popularity.reindex.interval` and `vsadmin.popularity.reindex.start` properties. Set to `false` if you do not want the recalculation process to run.

Note – If you have multiple Vending Managers sharing the same database, configure only one Vending Manager to recalculate the popularity ratings. Disable the function on all other Vending Managers.

1.10 Enabling Cleanup of Expired Items

Streamed content can have an end time assigned to it. Using custom fields, other content can also have an end time. You can configure Content Delivery Server to automatically deactivate items that have reached their expiration time so those items are no longer available to subscribers. To automatically set the status of expired content to `Inactive`, set the following properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/VSAdminConsole.properties` file:

- `vsadmin.expired_content.cleanup.enabled`. Indicates whether content is automatically deactivated. Set to `true` to automatically deactivate content. Set to `false` to prevent content from being automatically deactivated. The default is `true`.
- `vsadmin.expired_content.cleanup.interval`. Time, in seconds, between checks for expired content. The default is `3600`.

1.11 Configuring Device Client Web Services

Device Client web services are APIs that you can use over the web to access data in the Content Delivery Server database. You can use Device Client web services to authenticate a subscriber, discover, preview, purchase, and download content, access a subscriber's purchase history, and cancel a subscription to a content item.

If you plan to use the Device Client web services to create your own subscriber interface, set the following properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/SubscriberPortal.properties` file:

- `subscriberApi.authkey.timeout.minutes`. Number of minutes of inactivity after which an authentication key expires. The initial value is `10`.

- `webservices.v1.authenticate.headers.enable`. Flag that indicates if request headers can be used to authenticate a user. The request headers that are used depends on the `WAPGatewayAdapter` implementations that Content Delivery Server is configured to use. Set to `true` to enable the use of request headers. Set to `false` to require either anonymous use or user name and password authentication. The initial value is `false`.
- `webservice.v1.auto_provision.unknown.userpeer`. Flag that indicates if subscribers known to the external subscriber database, but not known to Content Delivery Server, are provisioned in Content Delivery Server. Set to `true` to provision unknown users. Set to `false` to prevent unknown users from being provisioned. The initial value is `true`. If set to `false`, a subscriber can be authenticated, but can use only Device Client web services that do not require a subscriber who is known to Content Delivery Server.

For information on Device Client web services, see Chapter 13 in the *Sun Java System Content Delivery Server Customization Guide*.

1.12 Configuring the Fulfillment Service

Requests for content using a URL from the device are handled through the Fulfillment Service. The MSISDN of the subscriber making the request is used to verify that the subscriber's subscriber plan includes the content being requested so the subscriber has access to the content. The MSISDN is retrieved from the request header. If the MSISDN is not part of the request header, verification fails and the content is not delivered.

If the request headers that are generated by requests from the devices that you support do not include the MSISDN, you must enable downloads to unprovisioned users to ensure that content is delivered. Allowing unprovisioned users to download content bypasses the verification of the subscriber plan so the MSISDN is not needed.

To enable downloads to unprovisioned users, set the `fs.allow_unprovisioned_downloads` property in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/FulfillmentService.properties` file to `true` for each Vending Manager deployment.

CHAPTER 2

Billing Integration

You do not need to change your billing implementation to use Content Delivery Server. You can configure Content Delivery Server to work with your current billing system through the use of billing adapters.

A billing adapter for postpaid or asynchronous billing converts the information provided by Content Delivery Server to the format needed by your billing system. Content Delivery Server posts billing events to a JMS queue. You can receive these billing events using a JMS client. The JMS client uses the billing adapter to format the information for your billing system.

A billing adapter for prepaid or synchronous billing is called by Content Delivery Server as the purchase is being processed. The adapter can dynamically change the price of content, if desired, validate the purchase in real time, or manage billing through an external system such as premium SMS.

You can create your own postpaid billing adapter using the Event Service API if the adapter provided does not meet your needs. You can create your own prepaid billing adapter using the Billing API. See the *Sun Java System Content Delivery Server Customization Guide* for information on these APIs.

This chapter includes the following topics:

- Billing Adapters Provided
- Working with the Postpaid Service
- Configuring External Content and Group IDs

2.1 Billing Adapters Provided

For postpaid billing, Content Delivery Server provides the Postpaid Service. This service includes a JMS client that processes the billing events in the event queue and generates a file that contains the information that your billing system can use to charge subscribers. The file format can be XML, comma-separated values (CSV), or name-value pairs. See Section 2.2, "Working with the Postpaid Service" on page 2-2.

No prepaid billing adapters are provided.

2.2 Working with the Postpaid Service

The Postpaid Service supports billing systems that charge subscribers after content has been purchased. You can use the Postpaid Service instead of a customized billing adapter if your billing system supports postpaid billing and processes records in one of the following formats:

- XML
- CSV
- Name-value pairs

2.2.1 Configure the Postpaid Service

To have Content Delivery Server support the postpaid billing model, configure the Postpaid Service. The handler used is the `PostpaidDefaultHandler`. To configure the Postpaid Service, follow these steps:

1. **Open the** `$CDS_HOME/deployment/`*deployment-name*`/conf/PostpaidService.properties` **file for edit.**

   Set the following properties:

   - `postpaid.handler.PostpaidDefaultHandler.events`. Specify the billing events that you want the handler to process. See the comments in the file for a list of valid events.
   - `postpaid.handler.PostpaidDefaultHandler.billingevent.process_free_downloads`. Set to `true` to process events for free content. Set to `false` if you do not want to process events for free content.

- `postpaid.handler.PostpaidDefaultHandler.billingevent.process_prepay_events`. Set to `true` to process events for prepaid content. Set to `false` if you do not want to process events for prepaid content.
- `postpaid.handler.PostpaidDefaultHandler.output.header`. To add a header to the top of the output file, see the comments in the file for information on setting this property.

2. **(Optional) Set up archive files for the billing records as desired:**
   - To prevent the generation of archive files and instead add billing records to a single file that grows until the service is stopped, set the following properties as shown:

```
postpaid.handler.PostpaidDefaultHandler.output.refresh.frequency=
postpaid.handler.PostpaidDefaultHandler.output.refresh.size=0
```

   - To create an archive file whenever a specific number of records are written, set `postpaid.handler.PostpaidDefaultHandler.output.refresh.size` to the number of records.
   - To create an archive file on a recurring basis, set the `postpaid.handler.PostpaidDefaultHandler.output.refresh.frequency` property to one of the following values:
     - `daily`. A new file is generated every day at the time the Postpaid Service started. For example, if the service started at 02:07:00, a new file is generated each day at 02:07:00.
     - `weekly`. A new file is generated on the same day of each week at the time that the Postpaid Service started. For example, if the service started on a Thursday at 22:30:57, a new file is generated every Thursday at 22:30:57.
     - `monthly`. A new file is generated on the same date each month at the time that the Postpaid Service started. For example, if the service started on March 14 at 21:23:34, a new file is generated on the 14th of each month at 21:23:34.
     - `yearly`. A new file is generated on the same date each year at the time the Postpaid Service started. For example, if the service started on January 6 at 04:10:05, a new file is generated every January 6th at 04:10:05.

Note – Stopping and restarting the Postpaid Service, restarts the period for which files are written. For example, if the frequency is set to weekly and the Postpaid Service is initially started on a Monday and then restarted on Thursday, the next file is written on the next Thursday, not on the next Monday.

---

3. **(Optional) Set the following properties to recover past billing records:**

   - `postpaid.handler.PostpaidDefaultHandler.recovery.enabled`. Set this property to `true` to enable the recovery of past billing records. Set to `false` if you do not want past billing records recovered.
   - `postpaid.handler.PostpaidDefaultHandler.recovery.starting.point`. Set this property to the timestamp for the beginning of the period for which you want to recover records. The value must be specified in the format `mm-dd-yyyy hh:mm:ss`, for example, `01-01-2004 00:00:01`.
   - `postpaid.handler.PostpaidDefaultHandler.recovery.stopping.point`. Set this property to the timestamp for the end of the period for which you want to recover records. The value must be specified in the format `mm-dd-yyyy hh:mm:ss`, for example, `01-01-2004 23:59:59`.
   - `postpaid.handler.PostpaidDefaultHandler.recovery.file.suffix`. Set this property to the string appended to the recovery file that is created. The default is `.recover`.

   If recovery is enabled, the next time the Postpaid Service is started, billing records for the period specified are written to the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/Postpaid.recover` file. When the recovery process completes, the file is renamed to `Postpaid.recover.`*`timestamp`*.

4. **To control the creation of the billing records file, set the `postpaid.handler.PostpaidDefaultHandler.output.refresh.empty_file` property.**

   The billing records file is generated at the following times:

   - The end of each period
   - When the Postpaid Service is restarted
   - When events are recovered

   Set the property to `true` to generate a file whether or not billing records exist. Set the property to `false` to generate a file only if billing records exist.

   The default if the property is missing is `true`. If the property is set to something other than `true` or `false`, `false` is assumed.

5. **To define the records that you want generated, set the `postpaid.handler.PostpaidDefaultHandler.output.template.file` property to the fully qualified name of the file that defines the records.**

   Use one of the following values:

   - `deployment/`*`deployment-name`*`/conf/resources/default_record.xsl`. Use this file with `PostpaidDefaultHandler` to generate name-value records.
   - `deployment/`*`deployment-name`*`/conf/resources/xml_record.xsl`. Use this file with `PostpaidDefaultHandler` to generate XML records.
   - `deployment/`*`deployment-name`*`/conf/resources/csv_record.xsl`. Use this file with `PostpaidCSVHandler` to generate CSV records.

6. **Save your changes to the `PostpaidService.properties` file.**

2.2.2 Billing Event Parameters

The following table shows the information provided for each billing event:

TABLE 2-1 Billing Event Parameters

Parameters	Description
`billing-ticket`	Billing ticket for this transaction.
`campaign_coupon`	Coupon code for a campaign.
`campaign_id`	String that identifies the campaign.
`catalog-res-id`	String that identifies the content edition.
`content_binary_mimetype`	MIME type of the content.
`content_class_id`	String that identifies the content item.
`content_description`	Long description of the content.
`content_drm_type_id`	String that identifies the DRM method used to protect the content.
`content_short_description`	Short description of the content.
`content-id`	String that identifies the content that was purchased. This value is the same as `catalog-res-id`.
`content_name`	Name of the content.
`current-status`	Current status of this transaction.
`date`	Date on which the transaction occurred.
`destination-address`	Address to which content is sent, for example, the MSISDN of the subscriber who requested content.
`developer-content-id`	Unique identifier used by the developer to identify the content.
`developer-id`	String that identifies the developer of the content.
`developer_name`	Name of the developer who submitted the content.
`download-confirm`	Flag that indicates whether a confirmation is required after a successful download.
`download-count`	Number of times the content can be downloaded for the price paid.
`download-current-count`	Number of times the subscriber has downloaded this content, including this time.

TABLE 2-1 Billing Event Parameters *(Continued)*

Parameters	Description
download-expiration	Flag that indicates whether the download period has expired.
download-period	Time period during which the content can be downloaded without additional charge to the subscriber.
download-price	Price of the content purchased.
download-purchase	Flag that indicates this is a purchase request.
download-recurring	Flag that indicates whether the subscriber is charged for each download.
event-log	Name of the event log.
event-msg	Message issued with the event.
event-source-type-id	Number that identifies the source of the event.
event-type	Numeric representation of the event that occurred.
event-type-id	String that identifies the type of event that occurred.
external_content_id	String that identifies the content to the billing system.
external_group_id	String that identifies the group to which the content belongs.
external-request-text	Text of the request from the subscriber, for example, the MO push request content.
gift_message	Message included with the gift.
gifted_current_downloads	Number of times the recipient downloaded this gift, including this time.
gifted_current_subscriptions	Number of subscription periods used by the recipient, including this period.
gift_download_date	Date that the gift was first downloaded by the recipient.
gift_expiration_date	Date by which the gift must be claimed by the recipient.
gift_purchase_date	Date the gift was purchased by the giver.
gifted_downloads	Number of downloads included in the gift.
gifted_subscriptions	Number of subscription periods included in the gift.
is_on_device	Flag that indicates whether the content is already on the device.
is-prepay	Flag that indicates whether the subscriber prepaid for the content.

TABLE 2-1 Billing Event Parameters *(Continued)*

Parameters	Description
`limited-time-end`	End date until which the content can be used.
`limited-time-price`	Price to use the content for a specified time period.
`limited-time-start`	Start date from which the content can be used.
`locale`	Subscriber's locale.
`msisdn`	MSISDN for the subscriber device.
`pricingoption_key`	String that identifies the pricing option.
`pricingoption_name`	Name of the pricing option.
`push-msgtext`	Message sent to the subscriber's device or email.
`recipient_locale_code`	Locale of the intended recipient of the content.
`recipient_login_id`	Login ID of the intended recipient of the content.
`recipient_mobile_id`	Mobile ID of the intended recipient of the content.
`recipient_unique_device_id`	Unique device ID of the intended recipient.
`server-id`	String that identifies the Vending Manager.
`session-id`	String that identifies the subscriber's session.
`source-address`	Address of the external entity from which the message was received, for example the MSISDN of the SMSC.
`subscription-expiration`	Date that the subscription period ends.
`subscription-frequency`	How often the subscription price is charged.
`subscription-recurring`	Flag that indicates whether to automatically charge the subscriber for the next period when the current subscription period ends.
`subscription-price`	Price of the subscription period.
`timestamp`	Time at which the transaction occurred.
`unique-device-id`	String that uniquely identifies the device used.
`usage-count`	Number of uses allowed for the price specified for `usage-price`.
`usage-price`	Price charged for the number of uses specified for `usage-count`.

TABLE 2-1 Billing Event Parameters *(Continued)*

Parameters	Description
`user-id`	String that identifies the user who initiated the transaction.
`username`	Login name for the subscriber.
`vending-res-id`	String by which the Vending Manager identifies the content.

2.3 Configuring External Content and Group IDs

If your billing system requires something other than Content Delivery Server content identifier to identify content, follow these steps to configure external content and group IDS for your system:

1. **In the** `CDS.properties` **file, set the** `common.external_content_id.enable` **property to** `true`**.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory. When this property is `true`, the administrator is prompted when stocking content to provide the content ID and group ID known to the billing system.

2. **Open the** `external_content_id_selection.xml` **file for edit.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf/resources` directory.

3. **Add an** `entry` **element for each known content ID under the** `content_id` **element, for example:**

```
<content_id>
    <entry default="true">ID-1A</entry>
    <entry>ID-1B</entry>
    <entry>ID-1C</entry>
</content_id>
```

   This list is provided to the administrator when external content IDs are assigned to content. For content that is auto-stocked, the external content ID is set to the value of the `entry` element that contains the attribute `default="true"`.

4. **Add an** `entry` **element for each known group ID under the** `group_id` **element, for example:**

```
<group_id>
    <entry>Games</entry>
    <entry>Pictures</entry>
</group_id>
```

This list is provided to the administrator when group IDs are assigned to content.

5. **Save your changes to the** `external_content_id_selection.xml` **file.**

CHAPTER 3

Subscriber Integration

Sun Java System Content Delivery Server uses an Oracle database to manage subscriber profiles. If you already have extensive subscriber data, you do not need to duplicate this information. Content Delivery Server can be configured to work with your existing subscriber data.

A subscriber adapter maps external subscriber data to the data required by Content Delivery Server when processing subscriber-related functions.

This chapter describes the subscriber adapter provided with Content Delivery Server. You can create your own subscriber adapter using the User Profile API. See the *Sun Java System Content Delivery Server Customization Guide* for information on this API.

This chapter includes the following topics:

- Subscriber Adapter Provided
- Using a Subscriber Adapter
- Working With LDAP
- Subscriber Account Management
- Automatic Provisioning of Subscribers

3.1 Subscriber Adapter Provided

Currently, the subscriber adapter provided with Content Delivery Server supports the Lightweight Directory Access Protocol (LDAP) format. The LDAP subscriber adapter uses an XML file to map data between Content Delivery Server and your LDAP directory.

3.2 Using a Subscriber Adapter

To specify the subscriber adapter that you want to use, you must set the `module.security.subscriber.usermanager` property to the fully qualified class name of the subscriber adapter. This property is in the `security.conf` file found in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory. Use one of the values shown in the following table.

Adapter	Value for the `module.security.subscriber.usermanager` Property
Oracle	`com.sun.content.server.vending.security.user.SubscriberImpl` Use this value to use Content Delivery Server database to store all subscriber data. This is the default setting.
LDAP	`com.sun.content.server.vending.security.user.ldap.ldapusermanager.LDAPUserManager` Use this value to use an external LDAP directory for subscriber data.

To use the subscriber adapter for LDAP, you must also provide an XML file that describes the mapping to be used. This file is described in Section 3.3.1, "Creating the Mapping File for LDAP" on page 3-2.

3.3 Working With LDAP

This section provides additional information on setting up your system to work with Content Delivery Server when your subscriber data is stored in an LDAP directory. Section 3.3.1, "Creating the Mapping File for LDAP" on page 3-2 describes how to create a file that maps the fields in the LDAP directory to the fields in Content Delivery Server. Section 3.3.2, "Configuring Access for Sun Java System Application Server" on page 3-8 describes changes that are needed to the security policy if you are using Sun Java System Application Server. Section 3.3.3, "Tuning LDAP" on page 3-8 describes how to set up LDAP to improve performance.

3.3.1 Creating the Mapping File for LDAP

To use subscriber data in an LDAP directory, you must create a mapping file in XML that maps the data needed by Content Delivery Server to the information in the LDAP directory. The `conf.xml` file in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory contains a sample mapping.

To create a mapping file:

1. **Copy the `conf.xml` file to a new file in the same directory, for example, `cdsmapping.xml`.**

2. **Edit your file to define the mapping of your LDAP data.**

   Section 3.3.1.1, "Subscriber Data for Content Delivery Server" on page 3-3 identifies the data that must be provided to Content Delivery Server. Section 3.3.1.2, "Sample Mapping File" on page 3-6 provides a sample file.

3. **Direct Content Delivery Server to use your file.**

   Set the `cds.security.ldapusermanager.config_file` property in the `$CDS_HOME/deployment/`*deployment-name*`/conf/security.config` file to the name of your file.

3.3.1.1 Subscriber Data for Content Delivery Server

The first few lines of the mapping file contain the connection information for the LDAP server. The following tables describe the properties in the mapping file.

TABLE 3-1 describes the configuration properties that must be set.

TABLE 3-1 Configuration Properties

Property	Description
`search_scope`	Scope of the search. Specify one of the following values: • `0` - Searches the named object. • `1` - Searches only one level of the named object. This is the default. • `2` - Searches the entire sub-tree of the named object.
`max_search_wait_time`	Maximum time in milliseconds that LDAP executes a search request. Use a negative value to indicate no limit.

TABLE 3-2 describes the properties that define your LDAP environment.

TABLE 3-2 LDAP Properties

Property	Description
`initial_context_factory`	Fully qualified class name of the initial context factory.
`provider_url`	URL of the provider (LDAP server). **Note -** If you are using Sun Java System Application Server, the URL must not contain spaces.
`prefix`	Prefix used.

TABLE 3-2 LDAP Properties *(Continued)*

Property	Description
`username`	User-distinguished name used to access LDAP.
`password`	Password associated with the user name.
`master_username`	User name for the master server. This property is optional.
`master_password`	Password associated with the master user name. This property is optional.
`object`	One or more objects from LDAP.

TABLE 3-3 identifies the subscriber data used by Content Delivery Server. Add an element with the field name in the XML file that you create. The required fields are noted.

TABLE 3-3 Subscriber Data

Content Delivery Server Field	Description
`loginId`	Login ID used by the subscriber to access the Subscriber Portal. **Note -** This field is required and must be mapped.
`password`	Password for the login ID provided. **Note -** This field is required and must be mapped.
`uniqueDeviceId`	Unique ID that identifies the subscriber by the device being used. Typically, this is the same as the MSISDN. **Note -** This field is required and must be mapped.
`firstName`	First name of the subscriber. **Note -** This field is required and must be mapped.
`middleName`	Middle initial of the subscriber.
`lastName`	Last name of the subscriber. **Note -** This field is required and must be mapped.
`gender`	Gender of the subscriber.
`street1`	Street address for the subscriber.
`street2`	Any additional address information required for the subscriber.
`city`	City information for the subscriber.
`state`	State information for the subscriber.
`postalcode`	Postal code for the subscriber.
`country`	Country where the subscriber resides.

TABLE 3-3 Subscriber Data *(Continued)*

Content Delivery Server Field	Description
`email`	Email address for the subscriber, used when sending password reminders or campaign notifications. **Note -** This field is required and must be mapped.
`phone`	Phone number for the subscriber.
`activatedate`	Date on which the subscriber account was activated.
`deactivatedate`	Date on which the subscriber account was deactivated.
`salutation`	Salutation by which the subscriber prefers to be addressed.
`enabled`	Status of the subscriber. If no value is provided, the default is enabled.
`msisdn`	TMSISDN number for the subscriber, used when sending messages to the subscriber's device. **Note -** This field is required and must be mapped.

The mapping is contained in the `<mapping>...</mapping>` section of the XML file. The mapping element has the following attributes:

- `isDeletable`. Set to `true` to allow user records to be deleted by Content Delivery Server. Set to `false` to prevent user records from being deleted.
- `isAddable`. Set to `true` to allow user records to be created by Content Delivery Server. Set to `false` to prevent user records from being created.

Each element in the mapping section can have one or more of the following attributes:

- `isRequired`. Set to `true` to indicate that the mapped field must not be null or empty. Set to `false` to indicate that the mapped field can be null or empty. If this attribute is set to `true` and a null or empty value is returned from the LDAP directory, an error message is generated.
- `isModifiable`. Set to `true` to allow the field to be modified by Content Delivery Server. Set to `false` to prevent the field from being modified.
- `isMultiple`. Set to `true` to indicate that more than one field in LDAP maps to the field in Content Delivery Server. If this attribute is true, you must include a `value`*n* element for each LDAP field, where *n* is a sequential number from 0 to *number of fields* - 1. For example, if the `uniqueDeviceId` field maps to `handsetID` and `mobileID`, you would add the following statements:

```
<uniqueDeviceID isMultiple="true">
   <value0>handsetID</value0>
   <value1>mobileID</value1>
</uniqueDeviceID>
```

The password element can also have the attribute `isEncoded`. Set this attribute to `true` if the password is stored as an encoded string. Set to `false` if the password is stored without encoding. The default is `false`.

See Section 3.3.1.2, "Sample Mapping File" on page 3-6 for an example.

3.3.1.2 Sample Mapping File

TABLE 3-4 describes sample data that is mapped in the sample mapping file that follows. Fields identified as having no mapping do not appear in the sample.

TABLE 3-4 Data for Sample LDAP File

Content Delivery Server Field	LDAP Field
`loginId`	`SSN`
`password`	`pwd`
`uniqueDeviceId`	`handsetID, mobileID`
`firstName`	`givenName`
`middleName`	(no mapping)
`lastName`	`familyName`
`gender`	(no mapping)
`street1`	`street`
`street2`	(no mapping)
`city`	`city`
`state`	(no mapping)
`postalcode`	`zipcode`
`country`	(no mapping)
`email`	`email`
`phone`	(no mapping)
`activatedate`	(no mapping)
`deactivatedate`	(no mapping)
`salutation`	(no mapping)
`enabled`	`status`
`msisdn`	`msisdn`

CODE EXAMPLE 3-1 Sample Mapping File for LDAP Data

```
<ldapusermanager>
  <config>
    <search_scope>1</search_scope>
    <max_search_wait_time>1000</max_search_wait_time>
  </config>
  <ldap>
    <initial_context_factory>com.sun.jndi.ldap.LdapCtxFactory
    </initial_context_factory>
    <provider_url>ldap://t1:389/ou=Users,o=LDAPUserManager
    </provider_url>
    <prefix>uid=</prefix>
    <username>cn=directory manager</username>
    <password>ldappwd</password>
    <master_username>cn=directory manager</master_username>
    <master_password>ldappwd</master_password>
  </ldap>
  <object>
    <obj0>top</obj0>
    <obj1>person</obj1>
    <obj2>organizationalPerson</obj1>
  </object>
  <mapping isDeletable="true" isAddable="true">
    <loginId isRequired="true">SSN</loginId>
    <password isRequired="true" isEncoded="false">pwd</password>
    <uniqueDeviceId isRequired="true" isModifiable="true" isMultiple="true">
      <value0>handsetID</value0>
      <value1>mobileID</value1>
    </uniqueDeviceId>
    <firstName isRequired="true">givenName</firstName>
    <lastName isRequired="true">familyName</lastName>
    <street1>street</street1>
    <city>city</city>
    <postalcode>zipcode<postalcode>
    <email isRequired="true">email</email>
    <enabled isRequired="true">status</enabled>
    <msisdn isRequired="true" isModifiable="true">msisdn</msisdn>
  </mapping>
</ldapusermanager>
```

3.3.2 Configuring Access for Sun Java System Application Server

If you are using Sun Java System Application Server, you must modify the security policy to grant access permission to classes used by the LDAP subscriber adapter. Edit the `server.policy` file located in the `$CDS_HOME/deployment/`*deployment-name*`/sun/domains/`*server-domain*`/config` directory and add the following statements:

```
grant {
    permission java.security.SecurityPermission "insertProvider.SunJSSE";
    permission java.lang.RuntimePermission "setFactory";
    permission java.net.NetPermission "specifyStreamHandler";
};
```

Restart Content Delivery Server after you save your changes.

3.3.3 Tuning LDAP

When using an LDAP directory as the subscriber database, you might want to create an index on any attribute mapped to the unique device ID, login ID, or MSISDN to improve performance. See the documentation for the LDAP directory that you are using for instructions on creating an index. Create the index on the attribute that is mapped to `uniqueDeviceId` in the mapping file that you created.

Creating an index is resource intensive and could affect system performance. Choose a time to create the index that is least likely to impact users.

3.4 Subscriber Account Management

A subscriber account contains information that identifies the subscriber and the device used to access Content Delivery Server. Subscriber accounts are managed by the Vending Manager administrator through the Vending Manager administration console. Subscribers can also access their own accounts through the Subscriber Portal.

You can configure the options presented to the subscriber and the administrator. For example, you can allow subscribers to only view their accounts, or prevent administrators from deleting accounts. You can also configure which fields

subscribers or administrators can edit. For example, you can prevent subscribers from changing their mobile phone numbers, or prevent administrators from changing the subscribers' names.

To specify the options that you want to make available and the fields that can be edited, edit the following files found in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory:

- `subsubmgr.xml`. Configure this file to specify the actions that the subscriber can take through the Subscriber Portal.
- `vsadminsubmgr.xml`. Configure this file to specify the actions that the administrator can take through the Vending Manager administration console.

The actions for managing subscriber accounts are add, edit, and delete. These actions are represented by the `<add>`, `<edit>`, and `<delete>` elements in the files. To allow an action, set the `isEnabled` attribute to `true`. To prevent an action, set the `isEnabled` attribute to `false`. The default is `false`. For example, to prevent subscribers from creating an account, include the following statement in the `subsubmgr.xml` file:

```
<add isEnabled=”false”/>
```

The elements within the <edit> element in each of the files show the fields that you can manage. Fields that are not represented by an element in the file, such as the login ID, are managed only by Content Delivery Server. Valid attributes for these fields are `isReadOnly` and `isRequired`. To prevent a field from being changed, set the `isReadOnly` attribute to `true`. To enable the field to be changed, set the `isReadOnly` parameter to `false`. The default is `false`.

Note – The `isRequired` attribute must be set to `true` for the `password`, `mobile_id`, `first_name`, `last_name`, `email`, and `status` elements.

The following code shows the relevant section of the default `subsubmgr.xml` file.

CODE EXAMPLE 3-2 Sample `subsubmgr.xml` File

```
<subscriber>
   <subscriber>
      <!-- For adding -->
      <add isEnabled="true"/>
      <!-- For editing -->
      <edit isEnabled="true">
         <password isRequired="true"/>
         <mobile_id isRequired="true" isReadOnly="true"/>

         <first_name isRequired="true"/>
         <middle_name/>
         <last_name isRequired="true"/>
```

CODE EXAMPLE 3-2 Sample `subsubmgr.xml` File *(Continued)*

```
            <salutation/>
            <gender/>

            <street_1/>
            <street_2/>
            <city/>
            <state/>
            <postal_code/>
            <country_code/>

            <email isRequired="true"/>
            <contact_phone/>
         </edit>
      </subscriber>
</subscriber>
```

The following code shows the relevant section of the default `vsadminsubmgr.xml` file.

CODE EXAMPLE 3-3 Sample `vsadminsubmgr.xml` File

```
<vsadmin>
   <subscriber>
      <!-- For adding -->
      <add isEnabled="true"/>
      <!-- For editing -->
      <edit isEnabled="true">
         <password isRequired="true"/>
         <status isRequired="true"/>
         <mobile_id isRequired="true"/>

         <first_name isRequired="true"/>
         <middle_name/>
         <last_name isRequired="true"/>
         <salutation/>
         <gender/>

         <street_1/>
         <street_2/>
         <city/>
         <state/>
         <postal_code/>
         <country_code/>

         <email isRequired="true"/>
         <contact_phone/>
      </edit>
      <!-- For deleting -->
```

CODE EXAMPLE 3-3 Sample `vsadminsubmgr.xml` File *(Continued)*

```
        <delete isEnabled="true"/>
    </subscriber>
</vsadmin>
```

3.5 Automatic Provisioning of Subscribers

To provision a subscriber is to register that subscriber and create an entry in the subscriber database. You can configure Content Delivery Server to automatically provision unregistered subscribers when they access the Subscriber Portal. If auto-provisioning is enabled and an unregistered subscriber attempts to download content, a subscriber account is created using the MSISDN number and the download is allowed.

The following properties are available in the `$CDS_HOME/deployment/`*deployment-name*`/conf/SubscriberPortal.properties` file for setting up auto-provisioning:

- `auto_provision.unknown.user` - Determines if unknown subscribers are automatically registered with Content Delivery Server. Set this property to `true` to automatically register unknown subscribers when they access Content Delivery Server. Set to `false` to prevent unknown subscribers from being registered automatically. The default is `false`.
- `auto_provision.unknown.userpeer` - Determines if subscribers can access Content Delivery Server without registering. Set this property to `true` to enable subscribers to access the system without registering with Content Delivery Server. Set to `false` to force subscribers to register before accessing the system. If set to `true` and `auto_provision.unknown.user` is also set to `true`, a subscriber account is automatically created for the unknown subscriber. The default is `true`.
- `auto_provision.unknown.userpeer.gifting` - Determines if unknown recipients of gifts or sharing are automatically registered. Set this property to `true` for Content Delivery Server to attempt to automatically create a subscriber account for an unknown subscriber to whom content has been gifted or shared. Set to `false` to prevent the automatic provisioning of the unknown recipient. The default is `true`.
- `auto_provision.unknown.userpeer.mobile_originated` - Determines if a new subscriber account is provisioned when a Mobile Originated (MO) request for content is received from an unregistered user. Set this property to `true` to provision an account when an MO request is received. Set to `false` to prevent the account from being provisioned. The default is `true`.

- `auto_provision.unknown.user.firstName` - Provides the value used for the subscriber's first name when provisioning an unknown user. The default is `External User`.
- `auto_provision.unknown.user.lastName` - Provides the value used for the subscriber's last name when provisioning an unknown user. The default is `External User`.
- `auto_provision.unknown.user.middleName` - Provides the value used for the subscriber's middle name when provisioning an unknown user. The default is `External User`.
- `auto_provision.unknown.user.enabled` - Specifies if the account created for an unknown user is enabled. Set the property to `true` to enable the account. Set to `false` to disable the account. The default is `true`.
- `auto_provision.unknown.user.email` - Provides the value used for the subscriber's email address when provisioning an unknown user. The default is `@external.com`.

CHAPTER 4

DRM Server Integration

One of the options provided with Content Delivery Server for protecting content requires an application that implements the Open Mobile Alliance (OMA) Digital Rights Management (DRM) 1.0 guidelines for managing digital rights. To use this option, you must provide the application with which Content Delivery Server works. Content Delivery Server provides support for SafeNet DRM Fusion Toolkit as the OMA DRM 1.0 implementation.

This chapter describes how to integrate Content Delivery Server with DRM Fusion Toolkit. For information on configuring DRM support in Content Delivery Server, see Chapter 13. For information on applying DRM to content, see the *Sun Java System Content Delivery Server Reference Manual*.

This chapter includes the following topics:

- Setting Up DRM Fusion Toolkit
- Configuring Content Delivery Server to Use DRM Fusion Toolkit

4.1 Setting Up DRM Fusion Toolkit

Obtain DRM Fusion Toolkit from SafeNet. See `http://www.safenet-inc.com/digital_rights_management/DRM_Fusion_Toolkit.asp` for information.

Install DRM Fusion Toolkit using the instructions provided with it. You can install this application on any server.

Note the directory path in which you installed DRM Fusion Toolkit, the Master Key Password that you specified during installation, and the name and location of the configuration Java Archive (JAR) file that is created. You need this information to configure Content Delivery Server to use DRM Fusion Toolkit as described in the next section.

4.2 Configuring Content Delivery Server to Use DRM Fusion Toolkit

After you install DRM Fusion Toolkit, you must set configuration properties for Content Delivery Server, copy the DRM Fusion Toolkit JAR files to your Content Delivery Server deployments, and ensure that OMA DRM 1.0 is enabled.

4.2.1 Setting the Configuration Properties

Set the properties shown in the following table. These properties are in the `$CDS_HOME/deployment/`*deployment-name*`/conf/drmfusion.properties` file. Set these properties in the Catalog Manager deployment and in each Vending Manager deployment.

Note – This file contains an unencrypted password. Set the file permissions to restrict access.

TABLE 4-1 Properties for DRM Fusion Toolkit

Property	Description
`protection.password`	Master Key Password for DRM Fusion Toolkit.
`protection.domain`	Domain name used to generate unique IDs for the protected content, for example, using `sun.com` as the domain generates IDs similar to `cid:2001-SD@sun.com`.

4.2.2 Copying DRM Fusion Toolkit JAR Files

The JAR files provided with DRM Fusion Toolkit must be copied to all Content Delivery Server deployments. These JAR files are found in the *safenet-home*`/modules` and *safenet-home*`/modules/third-party` directories, where *safenet-home* is the fully qualified name of the directory in which you installed DRM Fusion Toolkit. You must also copy the configuration JAR file `safenet-toolkit-configuration.jar` that was created when DRM Fusion Toolkit was installed. This file is in the same folder as the `install.sh` command for DRM Fusion Toolkit, typically *safenet-home*`/bin`.

Setting up Content Delivery Server with the DRM Fusion Toolkit JAR files depends on the application server you are using and whether you have already deployed. For each Content Delivery Server deployment, follow these steps:

1. **For all application servers, place all of the DRM Fusion Toolkit JAR files in the** `$CDS_HOME/dist/cds/lib/external` **directory.**

   The DRM Fusion Toolkit is now included in all future deployments.

2. **If you have existing deployments that need to use the DRM Fusion Toolkit, place the JAR files in the** `$CDS_HOME/deployment/`*deployment-name*`/lib/external` **directory for each deployment.**

   If you are using WebLogic Server, the classpath is handled for you.

   If you are using Sun Java System Application Server, update the classpath for each deployment:

   a. **Back up the** `$CDS_HOME/deployment/`*deployment-name*`/sun/domains/cdsdomain/config/domain.xml` **file before editing it so you can recover from any errors that might be introduced during editing.**

   b. **Edit** `domain.xml` **and modify the** `java-config` **element to add the absolute path for the JAR files to the** `classpath-suffix` **attribute.**

   c. **Save your changes.**

3. **Restart any existing deployment to make it aware of the new JAR files.**

4.2.3 Enabling OMA DRM 1.0

If you did not enable the OMA DRM 1.0 method when you deployed and configured Content Delivery Server, use the following command to enable this method:

```
cdsi db import [-conf db-configuration-file] -cs enableomadrm10.sqli
```

db-configuration-file is the name of the database configuration file that contains the information for creating the Catalog Manager schema. If *db-configuration-file* is not provided, the value specified for the `DEFAULT_DB` variable in the `init_env.sh` script is used. For information on enabling DRM methods, see Chapter 13.

CHAPTER 5

Streaming Configuration and Integration

Streaming is a technique for delivering content, typically video or audio, to a client that plays the content as it is received. When connected to a streaming server, Sun Java System Content Delivery Server can be used to accept, manage, and deliver both live and on-demand streamed content.

This chapter includes the following topics:

- Integrating With a Streaming Server
- Configuring Content Delivery Server for Streaming

Additional information on streaming is available in the *Sun Java System Content Delivery Server Reference Manual*.

5.1 Integrating With a Streaming Server

A streaming server is required to deliver streamed content to subscribers. Content Delivery Server accepts streamed content, manages the content in a similar manner as other content, and provides subscribers with access.

The streaming server interacts with the Vending Manager. When content is stocked, streamed content is copied to the streaming server. When a subscriber purchases streamed content, the purchase is authorized by the Vending Manager and the content is delivered by the streaming server.

If you have more than one Vending Manager, each Vending Manager can have a dedicated streaming server, or a streaming server can be shared by multiple Vending Managers.

A streaming adapter is used to integrate Content Delivery Server with the streaming server of your choice. You can write a streaming adapter using the Streaming API described in the *Sun Java System Content Delivery Server Customization Guide*. See the documentation for your streaming server for information on configuration properties that you might need to set.

5.2 Configuring Content Delivery Server for Streaming

Content Delivery Server needs to know the name of the streaming adapter and the host and port number of the streaming server. To provide this information, set the properties as described in the following steps:

1. **In the** `Streaming.properties` **file in the** `$CDS_HOME/deployment/`*deployment-name*`/conf` **directory, set the following properties:**
   - `streaming.adapter.impl.class` - Name of the implementation of the `StreamingAdapter` class that is your streaming adapter.
   - `streaming.server.host` - Host name or IP address of the server on which the streamed content is located.
   - `streaming.server.rtsp.port` - Port number for the server on which the streamed content is located.
2. **In the** `VSAdminConsole.properties` **file in the** `$CDS_HOME/deployment/`*deployment-name*`/conf` **directory, set the** `vsadmin.streaming.enable` **property.**

   Set this property to `true` to support streaming. Set to `false` if you do not want to support streaming.

CHAPTER 6

Single Sign-On Support

Sun Java System Content Delivery Server supports single sign-on. Single sign-on makes it possible for a subscriber who is signed on to an operator's service to access content provided by Content Delivery Server without having to sign on again.

6.1 Using the MSISDN Number Through the WAP Gateway

Single sign-on is achieved through the use of the Mobile Station Integrated Services Digital Network (MSISDN) number. This number is provided through the WAP gateway that you have configured Content Delivery Server to use (see Chapter 7). Content Delivery Server uses the MSISDN number to authenticate users.

PART II Messaging Configuration

This part of the *Sun Java System Content Delivery Server Integration and Configuration Guide* provides information on setting up your system to handle messages to and from Content Delivery Server.

This part contains the following chapters:

- WAP Gateway Configuration
- Outgoing Push Messages
- Mobile Originated Push Messages
- Messaging Service and Confirm Service Configuration

CHAPTER 7

WAP Gateway Configuration

A Wireless Application Protocol (WAP) gateway serves as a translator between web protocols and wireless protocols. You can configure Content Delivery Server to work with the WAP gateway of your choice.

A WAP gateway adapter parses the HTTP header from a WAP gateway to retrieve the MSISDN number, device profile, and other attributes needed by Content Delivery Server.

The WAP gateway adapters provided with Content Delivery Server are described in this chapter. You can create your own WAP gateway adapter using the WAP Gateway API. See the *Sun Java System Content Delivery Server Customization Guide* for information on this API.

This chapter contains the following topics:

- WAP Gateway Adapters Provided
- Using a WAP Gateway Adapter

7.1 WAP Gateway Adapters Provided

Content Delivery Server provides a default WAP gateway adapter that can be used with any WAP gateway that does not require the value returned for the unique ID or MSISDN to be parsed. In addition, adapters for the following WAP gateways are provided with Content Delivery Server:

- Nokia Activ Server 2.0.1
- Nokia Artus WAP Gateway
- Openwave WAP Gateway

7.1.1 Default WAP Gateway Adapter

The default WAP gateway adapter can be used with any WAP gateway that can use the unique ID or MSISDN value in the format that it is received. If the value must be parsed, you must use an adapter created specifically for the WAP gateway that you are using.

To use the default adapter, follow these steps:

1. **Set the** `default.unique.http_header.key` **property in the** `$CDS_HOME/deployment/`*deployment-name*`/conf/SubscriberPortal.properties` **file to the key used to retrieve the unique ID or MSISDN, for example:**

   ```
   default.unique.http_header.key=x-up-subno
   ```

2. **Make sure that the adapter is registered.**

   See Section 7.2, "Using a WAP Gateway Adapter" on page 7-3 for instructions.

7.1.2 Nokia Activ Server 2.0.1

The Nokia Activ Server WAP gateway adapter parses the HTTP headers from the Nokia Activ Server WAP gateway and passes the information to Content Delivery Server. To use this adapter, register the following class:

```
com.sun.content.server.common.gateway.nokia.NokiaActivServerWAPGateway
```

7.1.3 Nokia Artus WAP Gateway

The Nokia Artus WAP gateway adapter parses the HTTP headers from the Nokia Artus WAP gateway and passes the information to Content Delivery Server. To use this adapter, register the following class:

```
com.sun.content.server.common.gateway.nokia.NokiaArtusWAPGateway
```

7.1.4 Openwave WAP Gateway

The Openwave WAP Gateway parses the HTTP headers from the Openwave WAP gateway and passes the information to Content Delivery Server. To use this adapter, register the following class:

```
com.sun.content.server.common.gateway.openwave.OpenwaveWAPGateway
```

7.2 Using a WAP Gateway Adapter

To register the WAP gateway adapter that you want to use, add the class name to the `wapgateway.config` file in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory. For example, to register the default adapter, use the following statement:

```
module.gateway.id=com.sun.content.server.common.gateway.DefaultWAPGateway
```

Include only the names of the adapters that you want to use. Remove adapters that you are not using.

CHAPTER 8

Outgoing Push Messages

Push technology makes it possible for subscribers to receive the link to content without first having to request it from their device. Push technology can also be used to push content directly to a device. Sun Java System Content Delivery Server supports WAP push, SMS, and SMTP push formats for messages, and SMS and MMS push for content.

The Messaging Service pushes messages and content to subscribers. A push sender adapter serves as the interface between Content Delivery Server and your push implementation for delivering messages and content. Configure the Messaging Service to use the push sender adapters that you need.

The push sender adapters provided with Content Delivery Server are described in the following sections. If these adapters do not provide the functionality that you need, you can create your own push sender adapter using the Messaging API. See the *Sun Java System Content Delivery Server Customization Guide* for information on this API.

This chapter contains the following topics:

- Push Sender Adapters
- Using a Push Adapter
- Configuring the SMS Push Message
- Configuring Support for MMS

8.1 Push Sender Adapters

Content Delivery Server provides adapters for SMS, WAP, and SMTP push formats for messages. Several SMS formats are supported.

Delivery of binary content using either MMS or SMS is supported, but you must write your own adapter for each delivery method that you want to use.

8.1.1 SMS Push for Messages

Content Delivery Server supports the following SMS push protocols: Short Message Peer-to-Peer (SMPP) version 3.4, SMS HTTP, and Computer Interface to Message Distribution (CIMD2). Configure the Messaging Service for the protocol that you use.

8.1.1.1 Setting Up Support for SMPP

1. **Install the SMPP API library version 0.3.7 on the server on which the Messaging Service is running.**

   This library is available from SourceForge.net.

   a. **Download the file** `http://downloads.sourceforge.net/smppapi/smppapi-0.3.7.tar.gz`**.**

      Save this file in a temporary location.

   b. **Extract the file** `smppapi-0.3.7/lib/smppapi-0.3.7.jar` **from the file just downloaded.**

   c. **Copy the** `smppapi-0.3.7.jar` **file to the** `$CDS_HOME/deployment/`*deployment-name*`/lib/external` **directory.**

   d. **If you are using the Sun Java System Application Server, add** `$CDS_HOME/deployment/`*deployment-name*`/lib/external/smppapi-0.3.7.jar` **to the CLASSPATH using one of the following methods:**

      - Stop the server and edit the `$CDS_HOME/deployment/`*deployment-name*`/sun/domains/`*server-domain*`/config/domain.xml` file. Add the file to `classpath-suffix` attribute of the appropriate `java-config` element for the server.
      - Use the Sun Java System Application Server Administration Console to modify the Classpath Suffix of the settings for the Java virtual machine for the appropriate server.

2. **Open the** `MsgService.properties` **file.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory.

3. **Set the SMS properties as needed for your environment.**

   The following code shows the properties for a sample SMPP Push adapter.

```
smsc.hostname = 127.0.0.1
smsc.port = 11111
esme.system_id=user1
esme.password = usrpw
esme.system_type =
esme.destination=
smsc.gsm.ton = 2
smsc.gsm.npi = 0
```

4. **Set the push sender properties as needed for your environment.**

   The following code shows the properties for a sample SMPP Push adapter.

```
pushsender.send.keep_alive=true
pushsender.keep_alive.milliseconds=30000
```

5. **(Optional) Set the properties used to categorize the outgoing message from the server.**

   The SMSC can use these properties for billing or for other purposes. If a value is not set for a service type, the value of the `default.service_type` property is used.

```
subscriber_detail_url.service_type=
mobile_originated_detail_url.service_type=
gifting_detail_url.service_type=
content_sharing_detail_ur.service_type=
password_reminder.service_type=
campaign_message.service_type=
default.service_type=
```

6. **Save your changes to the `MsgService.properties` file.**

7. **Register your adapter in the `pushsenderfactory.xml` file.**

   When you have your own SMSC, specify the following class as your adapter in the `pushsenderfactory.xml` file as described in Section 8.2, "Using a Push Adapter" on page 8-6.

```
com.sun.content.server.smpp34impl.msgserver.push.SMSPushMsgSender
```

8.1.1.2 Setting Up Support for SMS HTTP

1. **Open the** `MsgService.properties` **file.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory.

2. **Set the SMS properties as needed for your environment.**

   The following code shows the properties for a sample SMS HTTP Push adapter.

```
# SMS HTTP properties
cds.sms.http.serverurl=
cds.sms.http.user=
cds.sms.http.password=
cds.sms.http.from=CDS
```

Note – For SMS HTTP, your parameter names might be different than the defaults provided. If the SMSC that you use requires parameters different than those supported by this adapter, you must write your own adapter using the Messaging API. For information on this API, see the *Sun Java System Content Delivery Server Customization Guide*.

   The following properties specify the names of the HTTP parameters to be passed in the SMS message.

```
cds.sms.attribname.userid
cds.sms.attribname.password
cds.sms.attribname.from
cds.sms.attribname.msg
cds.sms.attribname.to
```

   For example, if the SMSC uses the HTTP parameter `smsfrom` to identify where the message is from, set `cds.sms.attribname.from` to `smsfrom`. The values must not be null or blank.

3. **Save your changes to the** `MsgService.properties` **file.**

4. **Register your adapter in the** `pushsenderfactory.xml` **file.**

   When you are using HTTP for your SMS services, specify the following class as your adapter in the `pushsenderfactory.xml` file as described in Section 8.2, "Using a Push Adapter" on page 8-6.

```
com.sun.content.server.messagingservice.msgserver.push.HTTPSMSPushMsgSender
```

8.1.1.3 Setting Up Support for CIMD2

1. **Open the** `MsgService.properties` **file.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory.

2. **Set the SMS properties as needed for your environment.**

   The following code shows the properties for a sample CIMD2 Push adapter.

```
smsc.hostname = 127.0.0.1
smsc.port = 11111
esme.system_id=user1
esme.password = usrpw
```

3. **Save your changes to the** `MsgService.properties` **file.**

4. **Register your adapter in the** `pushsenderfactory.xml` **file.**

   When the SMSC that you use supports the CIMD2 protocol, specify the following class as your adapter in the `pushsenderfactory.xml` file as described in Section 8.2, "Using a Push Adapter" on page 8-6.

```
com.sun.content.server.messagingservice.msgserver.push.SMSCIMD2PushMsgSender
```

8.1.2 WAP Push for Messages

This adapter supports push delivery using WAP push. To configure the Messaging Service for WAP Push support, follow these steps:

1. **Open the** `MsgService.properties` **file.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory.

2. **Set the WAP and PPG properties as needed for your environment, for example:**

```
#WAP and PPG properties
asyncmsg.wap.ppg=
asyncmsg.wap.id=CDS
asyncmsg.wap.ip.bearer=
asyncmsg.wap.priority=high
asyncmsg.wap.bearer_type=SMS
asyncmsg.wap.bearer_required=false
asyncmsg.wap.delivery_method=unconfirmed
asyncmsg.wap.network_type=GSM
asyncmsg.wap.netwok_required=true
```

3. **Save your changes to the** `MsgService.properties` **file.**

4. **Register your adapter in the** `pushsenderfactory.xml` **file.**

   Use the class `com.sun.content.server.messagingservice.msgserver.push.WAPPushMsgSender` as your adapter. Specify this adapter in the `pushsenderfactory.xml` file as described in Section 8.2, "Using a Push Adapter" on page 8-6.

5. **If your WAP push proxy gateway (PPG) requires attributes other than those currently included in the message template, you must update the** `wap_push_msg_template.xsl` **file.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory.

8.1.3 SMTP Push for Messages

This adapter supports push delivery using SMTP. Use the class `com.sun.content.server.messagingservice.msgserver.push.SMTPPushMsgSender` as your adapter. Specify this adapter in the `pushsenderfactory.xml` file as described in Section 8.2, "Using a Push Adapter" on page 8-6.

8.1.4 SMS Push for Binary Content

If you want to push binary content to devices, you must create your own adapter using the Messaging API. Use the push category to determine whether binary content or a message is sent. The constant `PUSH_CONTENT_BINARY_CATEGORY` defined in the `PushConstants` class identifies messages that contain binary content. See the *Sun Java System Content Delivery Server Customization Guide* for information on the Messaging API and the `PushConstants` class.

If you write your own adapter, specify this adapter in the `pushsenderfactory.xml` file as described in Section 8.2, "Using a Push Adapter" on page 8-6.

8.2 Using a Push Adapter

The `pushsenderfactory.xml` file is used to register push sender adapters. As shown in the following code example, the `pushmsgsender` properties must include the fully qualified name of the push adapter class and the protocol that the adapter supports.

The following sample registers an adapter for each type of push sender supported.

CODE EXAMPLE 8-1 Sample `pushsenderfactory.xml` File

```
<pushmsgsenderset>
  <pushmsgsender0
    class=
"com.sun.content.server.messagingservice.msgserver.push.TestSMSPushMsgSenderImpl"
    protocol="sms"/>
  <pushmsgsender1
    class=
"com.sun.content.server.messagingservice.msgserver.push.WAPPushMsgSender"
    protocol="wap"/>
  <pushmsgsender2
    class=
"com.sun.content.server.messagingservice.msgserver.push.SMTPPushMsgSender"
    protocol="smtp"/>
  <pushmsgsender3
    class=
"com.sun.content.server.messagingservice.msgserver.push.MMSPushMsgSender"
    protocol="mms"/>
</pushmsgsenderset>
```

To specify the push sender adapters that you want to use, follow these steps:

1. **Register the adapters with Content Delivery Server.**

   To register the adapters, create an XML file named `pushsenderfactory.xml` in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory. Only one SMS push adapter can be specified.

   See CODE EXAMPLE 8-1 for an example of this file.

2. **Include the adapter class and any dependent classes in your classpath.**

3. **Save the `pushsenderfactory.xml` file.**

8.3 Configuring the SMS Push Message

The default SMS push message contains a message from Content Delivery Server and the URL for the content to be downloaded, for example:

```
Download: Application Name
http://servername:port/subscriber/main/ddd?subid=101&riid=115
```

If the length of the message sent causes a problem, you can limit this message to just the message from Content Delivery Server or just the URL. Edit the template `sms_push_msg_template.xsl` found in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory.

The following code shows a sample template.

```
1. <?xml version=”1.0” ?>
2. <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3. version=”1.0”>
4. <xsl:output method="text" indent="yes" />
5. <xsl:template match="/">
6. <xsl:value-of select="/SMSMessage/MESSAGE" />
7. <xsl:value-of select="/CDSSMSMessage/HREF" />
8. </xsl:template>
9. </xsl:stylesheet>
```

To limit the message to just the message from Content Delivery Server, remove line 7. To limit the message to just the URL, remove line 6.

8.4 Configuring Support for MMS

To have Content Delivery Server deliver content by MMS, you must configure the MMS messaging properties in the `VSAdminConsole.properties`, `SubscriberPortal.properties`, and `MsgService.properties` files. The Multimedia Message Service Center (MMSC) processes the MMS message based on the values that you enter for the properties. The MMS messaging properties and their values are described in TABLE 8-1.

1. **Edit the `VSAdminConsole.properties` and the `SubscriberPortal.properties` files.**

   These files are in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory.

2. **Set the following MMS messaging properties in each of the two files, so that the values match.**

The following code shows sample settings for MMS support.

```
mms_smil.template.filename=mms_smil_template.xsl
mms.message.class=AUTO
mms.message.priority=NORMAL
mms.message.sender.visibility=SHOW
```

```
mms.message.read_report_required=true
mms.message.delivery_report_required=true
admin.mms.from.address=address@host.com
```

3. **Set the following additional property in the** `SubscriberPortal.properties` **file, for example:**

   ```
   mms.fallback.pushtype=sms
   ```

   Note that the service that you specify must also be configured for Content Delivery Server. For more information on configuring SMS or WAP, see Chapter 10.

4. **Edit the** `MsgService.properties` **file.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory.

5. **Set the full class name for the MMS sender class, for example:**

```
mms.senderclass=
    com.sun.content.server.messagingservice.msgserver.push.TestMMSSenderImpl
```

The following table describes the properties for MMS messages.

TABLE 8-1 MMS Messaging Properties

MMS Property Name	Description
`admin.mms.from.address`	The email address or phone number used to send MMS content.
`mms.fallback.pushtype`	Identifies an alternate mechanism to push the message if MMS is not used. The following values are valid: • `sms` - Short message service (default) • `wap` - Wireless application protocol The alternate mechanism must be configured for Content Delivery Server. See Chapter 10 for more information.
`mms.message.class`	Provides the MMSC with categories for the messages. The following values are valid: • `ADVERTISEMENT` • `AUTO` (default) • `INFORMATIONAL` • `PERSONAL`

TABLE 8-1 MMS Messaging Properties *(Continued)*

MMS Property Name	Description
mms.message.delivery_report_required	Indicates if a delivery report is expected from the subscriber's device. The following values are valid: • true - Requires that a delivery report be received from the subscriber's device. This is the default. • false - Does not require a delivery report. For example, messages that are sent to subscribers as part of a promotional campaign could have this property set to false.
mms.message.priority	Provides the MMSC with priority levels for messages. The following values are valid: • HIGH • NORMAL (default) • LOW
mms.message.read_report_required	Indicates if a message indicating that the message has been read is expected from the subscriber's device. The following values are valid: • true - Requires that a read report be received from the subscriber's device. This is the default. • false - Does not require a read report. For example, messages that are sent to subscribers as part of a promotional campaign could have this property set to false.
mms.message.sender.visibility	Indicates if the name of the message sender is displayed or hidden. The following values are valid: • HIDE - The name of the sender is be displayed in the MMS message. • SHOW - The name of the sender is displayed in the MMS message. This is the default.
mms.senderclass	The full name of the class that implements the MMSSender class. The MMSSender class can be found in the cdsapi module.
mms_smil.template.filename	The name of the XSL template that is used to create the Synchronized Multimedia Integration Language (SMIL) data. The value of this property is expressed as the name of the template file and its path, relative to the conf directory.

CHAPTER 9

Mobile Originated Push Messages

Mobile Originated (MO) push technology makes it possible for subscribers to initiate requests for content. Push listener adapters are used by Content Delivery Server to receive these messages.

A push adapter serves as the interface between Content Delivery Server and your push implementation for receiving messages. Configure Content Delivery Server to use the push listener adapters that you need.

The push listener adapters provided with Content Delivery Server are described in the following sections. If these adapters do not provide the functionality that you need, you can create your own push listener adapter using the Messaging API. See the *Sun Java System Content Delivery Server Customization Guide* for information on this API.

This chapter contains the following topics:

- Push Listener Adapter
- Using a Push Adapter

9.1 Push Listener Adapter

Content Delivery Server provides push listener adapters for an SMSC that supports CIMD2 or SMPP 3.4. Messages directly from the SMSC in HTTP format are also supported.

Messages must be in the following form:

[*shared-keyword*] [*device-prefix* | *campaign-prefix*] *bundle-ID* | *content-ID* | *keyword* | *campaign-coupon*

TABLE 9-1 describes the message parameters.

TABLE 9-1 MO Push Message Parameters

Parameter	Description
shared-keyword	Identifier assigned by an SMSC. This parameter is required only if you use a shared keyword assigned by your SMSC.
device-prefix	String that identifies the type of device the subscriber is using. This parameter is required only for messages from users not registered with Content Delivery Server.
campaign-prefix	String that identifies the message as a request for a campaign.
bundle-ID	Bundle ID that identifies the bundle that the subscriber is requesting. Bundle IDs consist of a bundle prefix and a number assigned by Content Delivery Server when the bundle is created.
content-ID	Content ID that identifies the content that the subscriber is requesting. Content IDs are assigned by Content Delivery Server when content is submitted.
keyword	Keyword that identifies the content that the subscriber is requesting. Keywords are assigned by the Vending Manager administrator and are optional.
campaign-coupon	Coupon that is associated with the requested campaign. This parameter is valid only if *campaign-prefix* is used.

To handle MO push messages from subscribers, configure the Subscriber Portal and Messaging Service as described in the following sections.

9.1.1 Configuring the Subscriber Portal for MO Push

Set the following properties in the `SubscriberPortal.properties` file, which is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory:

- `mopush.use.shared_keyword` - Shared keyword assigned by the SMSC. The default is blank.
- `mopush.destination` - Short code or MSISDN to which messages are sent by the subscriber.
- `mopush.allow_web_requests` - Flag that indicates if instructions for requesting content are shown to unregistered users when using the PC-based Subscriber Portal. The default is `true`.
- `mopush.campaign.prefix` - Prefix used to identify a request for a campaign. The default is `c`.
- `mopush.nsm.device_prefix` - Prefix used to identify a device that uses the Nokia Smart Messaging (NSM) protocol to download content. The default is `n`.

- `mopush.nsm.user_agent` - User agent used with NSM. This property has no default and must be set before subscribers are given access to the system.
- `mopush.ems.device_prefix` - Prefix used to identify a device that uses the Enhanced Messaging Service (EMS) protocol to download content. The default is `e`.
- `mopush.ems.user_agent` - User agent used with EMS. This property has no default and must be set before subscribers are given access to the system.
- `mopush.wap.device_prefix` - Prefix used to identify a device that uses the OTA protocol to download content. The default is `w`.
- `mopush.wap.user_agent` - User agent used with WAP. This property has no default and must be set before subscribers are given access to the system.
- `mopush.accept.external.sms` - Flag that indicates if messages from any SMSC are accepted. Set to `true` to accept messages from any SMSC. Set to `false` to accept messages only from the SMSC specified in the `mopush.accept.moblieid` property.
- `mopush.accept.mobileid` - Phone number for the SMSC from which you want to accept messages. This number is used only if the `mopush.accept.external.sms` property is set to `false`.

Set the `bundle.prefix` property in the `CDS.properties` file, found in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory. This property defines the prefix used in a bundle ID. The default is `b`.

9.1.2 Configuring the Messaging Service for MO Push

Configure the Messaging Service for the formats that your SMSC supports. Adapters are provided for SMPP 3.4 and CIMD2. HTTP directly from the SMSC is also supported. If these formats do not meet your needs, you can write your own adapter using the Messaging API. See the *Sun Java System Content Delivery Server Customization Guide* for information on this API.

9.1.2.1 Setting Up Support for SMPP or CIMD2

1. **If not already installed, install the SMPP API library version 0.3.7 on the server on which the Messaging Service is running.**

   See Step 1 in Section 8.1.1.1, "Setting Up Support for SMPP" on page 8-2 for instructions.

2. **Set the `pushlistener.enable` property in the `MsgService.properties` file to `true`.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory.

3. **Save your change to the `MsgService.properties` file.**

4. **Set the properties in the `PushListener.properties` file for the connections that you want to support.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory. By default, a single connection is used to support all types of MO push requests. Set the `pushlistener.esme.system_id.all` and `pushlistener.esme.password.all` properties to the values required by your system.

CODE EXAMPLE 9-1 Single Connection for MO Push Requests

```
cds_mopush_action=all
pushlistener.esme.system_id.all =
pushlistener.esme.password.all =
pushlistener.cds_mopush_url.all = mo_push.do
```

If you prefer to support connections based on the type of data requested, follow these steps to configure the Messaging Service to support more than one connection to the SMSC:

a. **Remove support for the single connection to the SMSC.**

   Comment out the statements shown in CODE EXAMPLE 9-1 by adding a pound sign (#) to the beginning of each line.

b. **Add support for connections based on the type of data requested.**

   The following types of data are defined:

   - **Regular content** - Content that is delivered through the standard purchase process. To support a separate connection for regular content, uncomment the lines by removing the pound sign (#) at the beginning, and set the following system ID and password properties:

```
cds_mopush_action = regular_content
pushlistener.esme.system_id.regular_content =
pushlistener.esme.password.regular_content =
```

   - **Campaign** - Campaign information requested by the subscriber. To support a separate connection for campaigns, uncomment the lines by removing the pound sign (#) at the beginning, and set the following system ID and password properties:

```
cds_mopush_action = campaign
pushlistener.esme.system_id.campaign =
pushlistener.esme.password.campaign =
```

5. **Set the connection properties in the** `PushListener.properties` **file as needed for your environment.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory. These properties are used for all connections. The following code shows sample settings for SMPP.

```
pushlistener.smsc.hostname = 127.0.0.1
pushlistener.smsc.port = 11111
pushlistener.esme.system_id.all = user1
pushlistener.esme.password.all = usrpw
pushlistener.esme.system_type=
pushlistener.esme.destination=
pushlistener.smsc.gsm.ton = 2
pushlistener.smsc.gsm.npi = 0
pushlistener.send.keep_alive=true
pushlistener.keep_alive.milliseconds=30000
```

   The following code shows sample settings for CIMD2.

```
pushlistener.smsc.hostname = 127.0.0.1
pushlistener.smsc.port = 11111
pushlistener.esme.system_id.all = user1
pushlistener.esme.password.all = usrpw
```

6. **Save your changes to the** `PushListener.properties` **file.**

7. **Register your adapter in the** `pushlistenerfactory.xml` **file.**

   To use the SMPP adapter, specify the following class as your adapter in the `pushlistenerfactory.xml` file as described in Section 9.2, "Using a Push Adapter" on page 9-6.

```
com.sun.content.server.smpp34impl.msgserver.protocol.smpp.SMPPPushMsgListener
```

   To use the CIMD2 adapter, specify the following class as your adapter in the `pushlistenerfactory.xml` file as described in Section 9.2, "Using a Push Adapter" on page 9-6.

```
com.sun.content.server.messagingservice.msgserver.protocol.cimd2.CIMD2PushMsgL
istener
```

9.1.2.2 Setting Up Support for HTTP

1. **Set the** `pushlistener.enable` **property in the** `MsgService.properties` **file to** `false`**.**

   This file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory.

2. **Save your change.**

3. **Register the** `http://server:port/mo_push.do` **URL with your SMSC.**

   server is the name of the server on which the Subscriber Portal is deployed. *port* is the port number on which the Subscriber Portal listens for MO push messages. The values specified must be accessible by systems outside the network on which Content Delivery Server is running.

9.2 Using a Push Adapter

The `pushlistenerfactory.xml` file is used to register the push listener adapter that you choose to use. The `pushmsglistener` properties must include the fully qualified name of the push adapter class and the protocol that the adapter supports.

The following sample registers the adapter for CIMD2.

CODE EXAMPLE 9-2 Sample `pushlistenerfactory.xml` File

```
<pushmsglistenerset>
  <pushmsglistener0 class=
"com.sun.content.server.messagingservice.msgserver.protocol.cimd2.CIMD2PushMsg
Listener"
  protocol="sms"/>
</pushmsglistenerset>
```

To specify the push listener adapter that you want to use, follow these steps:

1. **Register the adapter with Content Delivery Server.**

   Create an XML file named `pushlistenerfactory.xml` in the `$CDS_HOME/deployment/`*deployment-name*`/conf` directory and set the properties as needed.

   See CODE EXAMPLE 9-2 for an example of this file.

2. **Include the adapter class and any dependent classes in your classpath.**

CHAPTER 10

Messaging Service and Confirm Service Configuration

The Messaging Service is used for sending messages to subscribers and listening for mobile originated (MO) messages. It also defines the Mail Service, which is used for campaign messages. The Confirm Service is used to handle confirmation messages sent after content is delivered to a device.

The procedures for setting up the Mail Service, storing response messages, and handling confirmation messages are described in the following sections.

This chapter contains the following topics:

- Configure the Mail Service
- Configure Storage of Response Messages
- Handling Confirmation Messages

10.1 Configure the Mail Service

Campaigns sent as email use the SMTP mail service defined for the Messaging Service. The mail service is initially configured when the Messaging Service is deployed based on properties for the Messaging Service that you set in the deployment configuration file.

To change the mail service used, modify the following properties in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/MsgService.properties` file:

- `mail.smtp.host`. Set this property to the name of the mail server that you want to use. The mail server must be capable of accepting unauthenticated email messages.
- `mail.smtp.sender`. Set this property to the reply-to address for email messages.

After you change the properties, restart the Messaging Service through the application server.

10.2 Configure Storage of Response Messages

For sent messages that contain binary content, Content Delivery Server receives responses from the SMSC or MMSC. To indicate the types of response messages to save in Content Delivery Server database, set the `store.message_response.push_types` property in the `MsgService.properties` file. The default is `sms,mms` to store responses from both SMS and MMS messages. This file is in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory.

10.3 Handling Confirmation Messages

To handle confirmation messages from an external entity after a successful download of content, follow these steps:

1. **Edit the `MsgService.properties` file.**

   This file is in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory.

2. **Set the `confirmlistener.enable` property to `true`.**

3. **Save your change.**

4. **Edit the `ConfirmListener.properties` file.**

   This file is in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory.

5. **If your system supports User Datagram Protocol (UDP), set the `udp.confirm_listener.port` property to the port number to which the device sends confirmation messages, for example:**

   ```
   udp.confirm_listener.port=2003
   ```

   This port number is specified by the device manufacturer and wireless carrier. The default is `2003`.

Note – To support MMS messages, you must write your own confirm service adapter. See the *Sun Java System Content Delivery Server Customization Guide* for information.

PART III Content Management

This part of the *Sun Java System Content Delivery Server Integration and Configuration Guide* provides information on setting up your system to manage your content.

This part contains the following chapters:

- Search Setup
- Pricing Configuration
- Digital Rights Management Configuration
- Submission Verifier Workflows
- Previews and Watermarks
- Content and Submission Formats
- Featured Content

CHAPTER 11

Search Setup

This release of Sun Java System Content Delivery Server provides expanded search functionality for all Content Delivery Server portals. Included in this functionality is the ability to specify the fields that are shown by default in the results and the fields that are searched by default. Custom fields can be included in the defaults.

This chapter includes the following topics:

- Configuring Default Result Fields
- Configuring the Search Engine

For an overview of search functionality, see the *Sun Java System Content Delivery Server Reference Manual*. For information on rebuilding search indexes, see the *Sun Java System Content Delivery Server System Management Guide*.

11.1 Configuring Default Result Fields

The following properties define the set of fields shown in the search results:

- `keyword.search.results` - Fields that are shown when the search query contains only keywords or specifies only fields in this list to be searched. These fields are also the fields that are shown when browsing by category or by status, therefore, the value should not be null.
- `field_query.search.results` - Fields that are shown when the search query specifies one or more fields to search that are not included in default results for keyword searches. Search and sort fields included in the query are shown in addition to the fields specified for this property.

- `user_defined.search.results` - Fields that are shown when the search query specifies one or more field names to include in the results. Result fields included in the query are shown in addition to the fields specified for this property. If the query specifies fields to search or on which to sort, the search and sort fields are also included in the results.

To specify more than one field for a property, separate the values with a comma (,). The fields are shown in the order in which the names are specified. Field names included in a query are added after the fields specified for the property. To be able to link to the content details from the search results, include either the title or short description in the results.

Note that specifying a large number of fields could slow performance. Also, if many fields are included in the results, horizontal scrolling might be needed when viewing the search results.

The following sections identify the files that contain these properties and the valid values.

11.1.1 Catalog Manager Administration Console

Properties for the Catalog Manager administration console are set in the `$CDS_HOME/deployment/`*deployment-name*`/conf/AdminConsole.properties` file. In addition to the properties described in Section 11.1, "Configuring Default Result Fields" on page 11-1, the following properties for specifying the fields shown when Vending Managers search the catalog on the Catalog Manager are also set in this file:

- `remote_vending.keyword.search.results` - Fields that are shown on the Vending Manager when the search query contains only keywords or specifies only fields in this list to be searched. These fields are also the fields that are shown when browsing by category, therefore, the value should not be null.
- `remote_vending.field_query.search.results` - Fields that are shown on the Vending Manager when the search query specifies one or more fields to search that are not included in default results for keyword searches. Search and sort fields included in the query are shown in addition to the fields specified for this property.
- `remote_vending.user_defined.search.results` - Fields that are shown on the Vending Manager when the search query specifies one or more field names to include in the results. Result fields included in the query are shown in addition to the fields specified for this property. If the query specifies fields to search or on which to sort, the search and sort fields are also included in the results.

Valid field names are shown in the following table. If you have defined custom fields, use `emf.`*custom-key* as the field name to include the field in the search results. See Section 1.8, "Setting Up Custom Fields" on page 1-10 for information on custom fields.

TABLE 11-1 Field Names for the Catalog Manager

Field Name	Description
`category`	Category name
`categoryid`	Category ID
`cmprice`	Catalog Manager price
`cmprice_enddate`	Last day that content can be downloaded
`cmprice_freqocc`	Frequency with which content can be downloaded
`cmprice_model`	Pricing model name
`cmprice_noofdays`	Number of days that content can be downloaded
`cmprice_nooftimes`	Number of times that content can be downloaded
`cmprice_startdate`	First day that content can be downloaded
`cmprice_value`	Monetary value of the Catalog Manager price
`ctype`	Content type
`devcontentid`	Developer content ID
`devid`	ID assigned by Content Delivery Server to a Developer Account
`devname`	Developer name
`dpprice`	Developer price
`dpprice_enddate`	Last day that content can be downloaded
`dpprice_freqocc`	Frequency with which content can be downloaded
`dpprice_model`	Pricing model name
`dpprice_noofdays`	Number of days that content can be downloaded
`dpprice_nooftimes`	Number of times that content can be downloaded
`dpprice_startdate`	First day that content can be downloaded
`dpprice_value`	Monetary value of the developer's price
`edname`	Edition name
`edstatus`	Edition status
`lastmoddate`	Date that the edition was last modified
`loctype`	Code that identifies the type and location of content
`longdesc`	Long description

TABLE 11-1 Field Names for the Catalog Manager *(Continued)*

Field Name	Description
matcheddevice	List of devices that are capable of running the content
matcheddeviceid	List of IDs of devices that are capable of running the content
maxsize	Size of edition binaries
planid	ID assigned by Content Delivery Server to a Developer Plan or Vending Plan
preview	Availability of preview
rcid	Resource Class ID
riid	Edition resource ID
shortdesc	Short description
status	Status
statusmsg	Status message
subdate	Submission date
title	Title
version	Edition version
weight	Edition weight

Default values are shown in the following code example. Note that property names and values must be on the same line, but the example is constrained by the size of the page.

```
keyword.search.results=
   title,status,devname,category,cmprice,dpprice,shortdesc,ctype
field_query.search.results=
   title,status,devname,category,cmprice,dpprice,shortdesc,ctype
user_defined.search.results=

remote_vending.keyword.search.results=
   title,status,devname,category,cmprice,shortdesc,ctype
remote_vending.field_query.search.results=
   title,status,devname,category,cmprice,shortdesc,ctype
remote_vending.user_defined.search.results=
```

11.1.2 Developer Portal

The properties described in Section 11.1, "Configuring Default Result Fields" on page 11-1, for the Developer Portal are set in the `$CDS_HOME/deployment/`*deployment-name*`/conf/DeveloperPortal.properties` file. Valid field names are shown in TABLE 11-1, with the exception of `devname`, which is not valid for the Developer Portal. If you have defined custom fields, use `emf.`*custom-key* as the field name to include the field in the search results. See Section 1.8, "Setting Up Custom Fields" on page 1-10 for information on custom fields.

Default values are shown in the following code example.

```
keyword.search.results=title,ctype,category,dpprice,statusmsg,status
field_query.search.results=title,ctype,category,dpprice,statusmsg,status
user_defined.search.results=
```

11.1.3 Vending Manager Administration Console

The properties described in Section 11.1, "Configuring Default Result Fields" on page 11-1, for the Vending Manager administration console are set in the `$CDS_HOME/deployment/`*deployment-name*`/conf/VSAdminConsole.properties` file. Valid field names are shown in the following table. If you have defined custom fields, use `emf.`*custom-key* as the field name to include the field in the search results. See Section 1.8, "Setting Up Custom Fields" on page 1-10 for information on custom fields.

TABLE 11-2 Field Names for the Vending Manager

Field Name	Description
`category`	Category Name
`categoryid`	Category ID
`citemid`	Category item ID
`cmprice`	Catalog Manager price
`cmprice_enddate`	Last day that content can be downloaded
`cmprice_freqocc`	Frequency with which content can be downloaded
`cmprice_model`	Pricing model name
`cmprice_noofdays`	Number of days that content can be downloaded
`cmprice_nooftimes`	Number of times that content can be downloaded
`cmprice_startdate`	First day that content can be downloaded
`cmprice_value`	Monetary value of the Catalog Manager price

TABLE 11-2 Field Names for the Vending Manager *(Continued)*

Field Name	Description
`ctype`	Content type
`ctype_concept`	ID assigned by Content Delivery Server to identify the type of content
`devname`	Developer name
`emf.popularity`	Popularity **Note -** See Section 1.9, "Defining Popularity" on page 1-14 for information.
`extcontgrpid`	External content group ID **Note -** Available only if external content and group IDs are enabled. See Section 2.3, "Configuring External Content and Group IDs" on page 2-8 for information.
`extcontid`	External content ID **Note -** Available only if external content and group IDs are enabled. See Section 2.3, "Configuring External Content and Group IDs" on page 2-8 for information.
`keyword`	Keyword used to identify content
`lastmoddate`	Date that the edition was last modified
`loctype`	Code that identifies the type and location of content
`longdesc`	Long description
`matcheddevice`	List of devices that are capable of running the content
`matcheddeviceid`	List of IDs of devices that are capable of running the content
`maxsize`	Size of edition binaries
`planid`	ID assigned by Content Delivery Server to a Subscriber Plan
`preview`	Availability of preview
`rcid`	Resource Class ID
`riid`	Edition resource ID
`shortdesc`	Short description
`status`	Status
`stockdate`	Date content was stocked
`title`	Title
`version`	Edition version
`vmprice`	Vending Manager price
`vmprice_enddate`	Last day that content can be downloaded

TABLE 11-2 Field Names for the Vending Manager *(Continued)*

Field Name	Description
vmprice_freqocc	Frequency with which content can be downloaded
vmprice_model	Pricing model name
vmprice_noofdays	Number of days that content can be downloaded
vmprice_nooftimes	Number of times that content can be downloaded
vmprice_startdate	First day that content can be downloaded
vmprice_value	Monetary value of the Vending Manager price

Default values are shown in the following code example.

```
keyword.search.results=rcid,title,status,vmprice,cmprice,category,ctype
field_query.search.results=rcid,title,status,vmprice,cmprice,category,ctype
user_defined.search.results=
```

11.1.4 Subscriber Portal

The properties described in Section 11.1, "Configuring Default Result Fields" on page 11-1 do not apply to the Subscriber Portal. Search results are returned to the Subscriber Portal through the Subscriber API in an `IContentSummary` object. Only fields included in this object are available for display. See the output of the Javadoc tool in the `$CDS_HOME/javadoc/subscriberapi` directory for information on this interface.

11.2 Configuring the Search Engine

Content Delivery Server uses the Solr search server to index content and handle search queries. Fields that are included in the index are defined in a file named `schema.xml`. The Catalog Search Service and the Vending Search Service each has its own copy of this file.

For the Catalog Search Service, the `schema.xml` file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf/css/solr/conf` directory. This file is used for searches in both the Catalog Manager and the Developer Portal.

For the Vending Search Service, the `schema.xml` file is in the `$CDS_HOME/deployment/`*deployment-name*`/conf/vss/solr/conf` directory. This file is used for searches in both the Vending Manager and the Subscriber Portal.

11.2.1 Configuring the Default Search Fields

Default search fields are the fields that are searched if the search query does not identify the fields to be searched. Because the schema file for the Catalog Search Service is shared between the Catalog Manager and the Developer Portal and the schema file for the Vending Search Service is shared between the Vending Manager and the Subscriber Portal, make sure that the default search fields that you specify are common to both portals.

Initially, the following fields are the default search fields for both the Catalog Search Service and the Vending Search Service:

- `ctype` - Content type
- `title` - Title
- `shortdesc` - Short description
- `longdesc` - Long description
- `category` - Category
- `devname` - Developer name
- `status` - Status

To change the default search fields, follow these steps:

1. **Open the `schema.xml` file for the search service that you are setting up.**

   For the Catalog Search Service, this file is in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/css/solr/conf` directory. For the Vending Search Service, this file is in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/vss/solr/conf` directory.

2. **Find the `defaultSearchField` element.**

   This element defines `text` as the default field to be searched when no field is specified. Fields included in this default field are identified by the `copyField` elements where the attribute `dest` is set to `text`, for example:

   ```
   <copyField source="ctype" dest="text"/>
   ```

3. **Add `copyField` elements for the fields that you want included in the default search.**

   For each field, set `source` to the name of the field and set `dest` to `text`. Valid field names for the Catalog Search Service are identified in TABLE 11-1. Valid field names for the Vending Search Service are identified in TABLE 11-2. If you have defined custom fields, use `emf.`*`custom-key`* as the field name.

4. **Remove `copyField` elements for fields that you do not want included in the default search.**

5. **Save your changes.**

6. **If the search service is deployed on a separate server, copy the `schema.xml` file to the server on which the Catalog Manager or Vending Manager is deployed.**

   For the Catalog Manager, copy the `schema.xml` file from the Catalog Search Service. For the Vending Manager, copy the `schema.xml` file from the Vending Search Service.

7. **Restart the search service to make the changes available.**

   Depending on the search service for which changes were made, restart the Catalog Search Service or the Vending Search Service.

11.2.2 Adding Custom Fields to the Search Index

If you defined custom fields for your system and you want users to be able to search these fields, you must add them to the search index. The search engine uses the information in the `schema.xml` file to create the index for each search service.

To add custom fields to the search index, follow these steps:

1. **Open the `schema.xml` file for the search service that you are setting up.**

   For the Catalog Search Service, this file is in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/css/solr/conf` directory. For the Vending Search Service, this file is in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/vss/solr/conf` directory.

2. **Find the `fields` element.**

   This element contains the `field` elements that identify the fields that are indexed by the search engine, for example:

```
<field name="ctype" type="text" indexed="true" stored="true"/>
```

3. **Add a `field` element for each custom field that you want included in the search index.**

   The following table identifies the attributes for the `field` element.

Attribute	Description
`name`	Name of the custom field in the format `emf.`*custom-key*. See Section 1.8, "Setting Up Custom Fields" on page 1-10 for information.
`type`	Data type for the field. The following list shows the value to enter depending on the data type specified for the custom field. • For custom field type `number`, use `sfloat` • For custom field type `text`, use `text`. • For custom field type `timestamp`, use `date`. • For custom field type `boolean`, use `boolean`.
`indexed`	Flag that indicates if the field is to be searched. Set to `true` to include the custom field in searches.
`stored`	Flag that indicates if the field is retrievable. Set to `true` to retrieve the custom field.
`multiValued`	Flag that indicates if the field can have multiple values. Set to `true` if the scope of the custom field is `edition`.

   For example, the following statement adds the custom field for artist to the search schema:

```
<field name="emf.artist" type="text" indexed="true" stored="true"/>
```

   If the custom field is not defined as either viewable or editable for the Catalog Manager or the Developer Portal, do not include the field in the `schema.xml` file for the Catalog Search Service. If the custom field is not defined as either viewable or editable for the Vending Manager or the Subscriber Portal, do not include the field in the `schema.xml` file for the Vending Search Service.

4. **(Optional) Enable the field for sorting.**

   If the scope of the custom field is `item` and the data type is `text`, follow these steps to enable the search results to be sorted on this field:

   a. **Add another field element using the custom field name followed by the suffix `_exact` for the `name` attribute.**

      Set the `type` attribute to `nonTokenizedSort`, the `indexed` attribute to `true`, and the `stored` attribute to `false`. For example, the following statement adds a sortable field for the custom field for artist:

```
<field name="emf.artist_exact" type="nonTokenizedSort" indexed="true" stored=
"false"/>
```

b. **Add a `copyField` element after the field definitions to copy the custom field to the field just defined.**

   For example, the following statement copies the custom field for artist to the `emf.artist_exact` field created in the previous step:

   ```
   <copyField source="emf.artist" dest="emf.artist_exact"/>
   ```

5. **To include a custom field in the default search fields, see** Section 11.2.1, "Configuring the Default Search Fields" on page 11-8**.**

6. **Save your changes.**

7. **If the search service is deployed on a separate server, copy the `schema.xml` file to the server on which the Catalog Manager or Vending Manager is deployed.**

   For the Catalog Manager, copy the `schema.xml` file from the Catalog Search Service. For the Vending Manager, copy the `schema.xml` file from the Vending Search Service.

8. **Set the column title shown for the custom field in the search results.**

   The column title is set for each portal in the following property files, which are located in the `$CDS_HOME/deployment/`*deployment-name*`/localization` directory:

   - `AdminConsoleMessages.properties`
   - `DevPortalMessages.properties`
   - `VendingManagerMessages.properties`
   - `SubscriberPortalLocaleResources.properties`

   For each portal in which the field appears, set the `search.column.title.emf.`*custom-key* property to the column title shown for the custom field when search results are displayed. Typically, this is the same string that is used for the `emf.`*content-type*`.`*custom-key*`.label` property described in Section 1.8.2, "Assigning Labels and Hints" on page 1-13.

9. **Restart the search service to make the changes available.**

   Depending on the search service for which changes were made, restart the Catalog Search Service or the Vending Search Service.

Note – Content that uses the new fields might be submitted between the time that you define the fields and the time that you add the fields to the schema file. If so, after updating the search schema, manually rebuild the search indexes to ensure that they contain the correct data. See Section 1.4 in the *Sun Java System Content Delivery Server System Management Guide* for information on rebuilding the search indexes.

CHAPTER 12

Pricing Configuration

The default currency in which content is priced is United States dollars. The pricing of content is managed by the Catalog Manager administrator and the Vending Manager administrator. However, to use a currency other than United States dollars, you must change the settings of the currency properties.

Pricing models define how subscribers pay for content, for example, by number of uses or uses in a period of time. Pricing models are managed by the Catalog Manager administrator. However, to add a grace period to content charged on every download, you must define the grace period.

This chapter includes the following topics:

- Set the Currency Symbol
- Set a Grace Period

12.1 Set the Currency Symbol

The currency used in Content Delivery Server system is defined in the `CDS.properties` file in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory. Only one currency is specified per deployment and that currency is used for all components in that deployment. If you deployed multiple Vending Managers, you can specify a different currency for each one as needed.

To specify the currency to use, set the following properties in the `CDS.properties` file for the deployment.

TABLE 12-1 Currency Properties

Property	Description
`i18n.currency.locale`	Locale code for the currency to use, for example, `fr_CA` for French Canadian.
`i18n.currency.symbol`	Characters used for the monetary symbol, for example, `$` for dollars.
`i18n.currency.code`	Code used for the currency, for example, `CAD` for Canadian dollars.
`i18n.currency.position`	Position of the symbol with respect to the units. Valid values are `before` and `after`.

Note – Changing the currency settings does not convert the amounts already in the system. If a price was entered as $1.50 when the settings were set for United States dollars, the price is £1.50 after the settings are changed to English pounds.

Vending Managers that are deployed separately from the Catalog Manager have an additional set of currency properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/VSAdminConsole.properties` file. These properties identify the currency used by the Catalog Manager. A Vending Manager that is deployed separately from the Catalog Manager can use a different currency. A Vending Manager that is deployed with the Catalog Manager must use the same currency as the Catalog Manager.

Settings for the following properties in the `VSAdminConsole.properties` file must match the corresponding settings in the `CDS.properties` file for the Catalog Manager deployment. Otherwise, incorrect catalog prices are shown in the Vending Manager administration console.

TABLE 12-2 Catalog Manager Currency Properties

Property	Description
`i18n.currency.locale`	Locale code for the currency used by the Catalog Manager, for example, `en_US` for American English.
`i18n.currency.symbol`	Characters used for the monetary symbol, for example, `$` for dollars.

TABLE 12-2 Catalog Manager Currency Properties *(Continued)*

Property	Description
`i18n.currency.code`	Code used for the currency, for example, `USD` for United States dollars.
`i18n.currency.position`	Position of the symbol with respect to the units. Valid values are `before` and `after`.

12.2 Set a Grace Period

A grace period provides subscribers with additional downloads of the same content at no charge. You might want to set a grace period if subscribers are known to have difficulty downloading content on the first try. The grace period can be a set number of downloads, a set number of days during which unlimited downloads are allowed, or a combination of downloads and days. If a combination is used, the grace period ends when one of the two conditions is met. For example, a grace period of five downloads and two days ends when the subscriber downloads the content five times or when two days pass, whichever occurs first.

Grace periods apply only to content associated with the Every Download pricing model. See the *Sun Java System Content Delivery Server Reference Manual* for information on pricing models. The default values indicate no grace period and subscribers are charged for every download.

To provide a grace period, set the following properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/CDS.properties` file:

- `pricing.model.recurringDownload.numberOfDays`. Set this property to the number of days that unlimited downloads are allowed per purchase. The default is `0`.
- `pricing.model.recurringDownload.numberOfTimes`. Set this property to the number of downloads allowed per purchase.The default is `1`.

CHAPTER 13

Digital Rights Management Configuration

Digital rights management (DRM) protects content against unauthorized use and distribution. This section describes how to set up Content Delivery Server for the types of DRM that you want to support. If you do not want to use the DRM feature, disable support for all DRM types.

This chapter includes the following topics:

- Setting the DRM Methods Supported
- Setting the Preferred Delivery Type
- Setting the Preferred Action for Devices that Do Not Support OMA DRM 1.0
- Setting the Event Handler

The properties described in this chapter need to be set only in the Catalog Manager deployment.

13.1 Setting the DRM Methods Supported

In Content Delivery Server, the DRM method used for an item of content is determined by the content type. The pricing models available for an item of content are determined by the DRM method assigned to the content type.

The Catalog Manager administrator is responsible for assigning DRM methods to content types and for pricing content. Work with the administrator to ensure that you properly configure the system to support the needs of your business. See the *Sun Java System Content Delivery Server Reference Manual* for more information on DRM.

Before you start using Content Delivery Server, you must enable the DRM methods that you want to support and disable the methods that you do not want to support. The status of each DRM method is stored in the Catalog Manager database, so this configuration is done only once on the server on which the Catalog Manager is deployed.

Content Delivery Server supports the following DRM methods:

- **None.** No protection is applied. Use this option when you do not need to control distribution or enforce usage rights, or if you are submitting content that is prewrapped with OMA DRM protection. This option cannot be disabled.
- **CDS DRM Agent.** Content is protected by one of the Content Delivery Server Digital Rights Management (CDS DRM) agents described in TABLE 13-2. Use this method to protect Java technology-based applications (Java applications) without requiring additional DRM software. Only one CDS DRM agent can be used per deployment. See Section 13.1.1, "Content Delivery Server DRM Agents" on page 13-3 for information on specifying the agent used.
- **CDS OMA DRM 1.0 Forward Lock.** Content is protected by the Content Delivery Server implementation of the guidelines defined by OMA DRM 1.0 for preventing content from being forwarded to other devices after purchase. Use this method to prevent unauthorized redistribution of content without requiring additional DRM software.
- **OMA DRM 1.0.** Content is protected according to the guidelines defined by OMA. See Section 13.1.2, "OMA DRM 1.0 Methods" on page 13-5 for additional information. Use this method to prevent unauthorized use and redistribution of content. You must have an application that implements the OMA DRM 1.0 guidelines available to handle the rights management when using this DRM type. To use DRM Fusion Toolkit from SafeNet, see Chapter 4.

Use the following command to enable and disable DRM methods:

`cdsi db import [-conf` *db-configuration-file*`] -cs` *file*

db-configuration-file is the name of the database configuration file that contains the information for creating the schemas. If *db-configuration-file* is not provided, the value specified for the `DEFAULT_DB` variable in the `init_env.sh` script is used. The following table describes the valid values for *file*.

TABLE 13-1 Files for Enabling and Disabling CDS DRM Agents

Script	Action
`enablecdsdrmagent.sqli`	Enables the CDS DRM Agent method.
`disablecdsdrmagent.sqli`	Disables the CDS DRM Agent method.
`enablecdsdrmfl.sqli`	Enables the CDS OMA DRM 1.0 Forward Lock method.

TABLE 13-1 Files for Enabling and Disabling CDS DRM Agents *(Continued)*

Script	Action
`disablecdsdrmfl.sqli`	Disables the CDS OMA DRM 1.0 Forward Lock method.
`enableomadrm10.sqli`	Enables the OMA DRM 1.0 method.
`disableomadrm10.sqli`	Disables the OMA DRM 1.0 method.

If you enable the CDS DRM Agent method, you must specify the agent to use. See the next section for instructions.

Note – Before you disable a DRM method, make sure that method is not assigned to a content type. Submission of content fails if the type of content submitted is protected by a disabled DRM method.

13.1.1 Content Delivery Server DRM Agents

CDS DRM agents are provided with Content Delivery Server for use in protecting Java applications. These agents contain the information needed to enforce the usage rights purchased by a subscriber. No additional DRM software is needed. When content is submitted, the agent that you choose to use is added to the content, which increases the size of the content by the average amount stated in the table. The following table describes the CDS DRM agents.

TABLE 13-2 CDS DRM Agents

Title	Description	Size (average)
Disconnected Time	Supports the download pricing model and the time-based (recurring subscription and per period) pricing models. When instrumented content is run, the license is validated without contacting Content Delivery Server. When the license expires, the subscriber must return to the Subscriber Portal to purchase another period of time and download the application again.	3.2 kilobytes
Disconnected Use	Supports the download pricing model and the usage-based (trial and per use) pricing models. When instrumented content is run, the license is validated without contacting Content Delivery Server. When the license expires, the subscriber must return to the Subscriber Portal to purchase more uses and download the application again.	4.4 kilobytes
Disconnected Use and Time	Supports all pricing models. When instrumented content is run, the license is validated without contacting Content Delivery Server. When the license expires, the subscriber must return to the Subscriber Portal to purchase either another period of time or more uses, and download the application again.	4.6 kilobytes

TABLE 13-2 CDS DRM Agents *(Continued)*

Title	Description	Size (average)
Small Connected Use and Time	Supports all pricing models. When instrumented content is run, Content Delivery Server is contacted to validate the license. When the license expires, the subscriber can renew the license and does not need to download the application again. **Note -** To prevent runtime errors, do not use this DRM agent with applications that perform any of the following actions: • Show an alert before calling the `startApp` method. • List their Record Stores. • Modify or remove the Content Delivery Server Record Store.	5.2 kilobytes
Medium Connected Use and Time	Supports all pricing models. When instrumented content is run, Content Delivery Server is contacted to validate the license. When the license expires, the subscriber can renew the license and does not need to download the application again. **Note -** To prevent runtime errors, do not use this DRM agent with applications that perform any of the following actions: • Show an alert before calling the `startApp` method. • Modify or remove the Content Delivery Server Record Store.	5.8 kilobytes
Standard Connected Use and Time	Supports all pricing models. When instrumented content is run, Content Delivery Server is contacted to validate the license. When the license expires, the subscriber can renew the license and does not need to download the application again.	7.0 kilobytes

To specify the CDS DRM agent that you want to use, set the `DRMAgent` property in the `$CDS_HOME/deployment/`*deployment-name*`/conf/cdsdrmagent.properties` file to one or more of the following values:

- `DisconnectedTime`
- `DisconnectedUse`
- `DisconnectedUseTime`
- `SmallConnectedUseTime`
- `MediumConnectedUseTime`
- `StandardConnectedUseTime`

If more than one DRM agent is specified, separate the agents with a comma, for example:

```
DRMAgent=DisconnectedTime,SmallConnectedUseTime
```

You can also specify multiple instances of the `DRMAgent` property, if preferred, with each instance set to a different agent. In both cases, a derived edition is created for each DRM agent in the order in which the agents are specified. To have an edition with a smaller agent available only if the edition with the larger agent is too large, specify the DRM agents in order from smallest to largest.

The `DRMAgent` property needs to be set only for the deployment that contains the Catalog Manager.

13.1.2 OMA DRM 1.0 Methods

Content Delivery Server supports the following DRM methods described by the Open Mobile Alliance (OMA) DRM 1.0 specification:

- **Forward Lock -** This method prevents content from being forwarded to other devices. Usage rights are not enforced with this method.
- **Combined Delivery** - This method includes a digital rights object with the content. The rights object prevents content from being forwarded to other devices and enforces usage rights.
- **Separate Delivery** - This method includes a digital rights object that is delivered separately from the content. Separate delivery allows content to be forwarded to other devices, which then access the separate rights object for permission to use the content. Not all devices support separate delivery.

OMA DRM 1.0 can be used for all types of content. Use of this method requires an external DRM server. To use DRM Fusion Toolkit from SafeNet, see Chapter 4. Separate delivery requires an SMSC or WAP PPG that can handle the rights object.

13.2 Setting the Preferred Delivery Type

The OMA DRM 1.0 specification provides for the delivery of the digital rights object either with the content (combined delivery) or separate from the content (separate delivery). Not all devices support separate delivery. If your system supports the OMA DRM 1.0 method, you must choose the preferred delivery method for the rights associated with the content.

In the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/omadrm10.properties` file, set the `oma.drm10.rights` property to one of the values described in the following table. This property needs to be set only for the deployment that contains the Catalog Manager.

TABLE 13-3 Delivery Type Settings

Setting	Description
CD	Combined delivery is used to deliver the rights with the content. With this setting, content protected using either combined delivery or separate delivery is made available to subscribers with devices that support either type of delivery.
SD	Separate delivery is used to deliver the rights separate from the content. If a subscriber's device does not support separate delivery, the content protected using separate delivery is not made available to that subscriber.
SD_CD	Separate delivery is the preferred delivery method. With this setting, content protected using either combined delivery or separate delivery is made available to subscribers with devices that support either type of delivery. Combined delivery is used for devices that do not support separate delivery.

13.3 Setting the Preferred Action for Devices that Do Not Support OMA DRM 1.0

Not all devices are compliant with OMA DRM 1.0. Content protected by either combined delivery or separate delivery cannot be delivered to non-compliant devices. For content protected by Forward Lock, you must choose whether or not that content is available to subscribers with non-compliant devices.

In the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/omadrm10.properties` file, set the `oma.drm10.plain` property to one of the values described in the following table. This property needs to be set only for the deployment that contains the Catalog Manager.

TABLE 13-4 Settings for Non-Compliant Devices

Setting	Description
ALL	Content is delivered without protection to subscribers with non-compliant devices.
NONE	Content is not available to subscribers with non-compliant devices.
FREE	Only free content is delivered without protection to subscribers with non-compliant devices.

13.4 Setting the Event Handler

Event handlers listen for specific events generated by the Event Service and respond to the events as needed. Support for separate delivery requires the use of an event handler. Content Delivery Server provides an event handler that you can use with SafeNet DRM Fusion Toolkit or any other DRM server.

To enable the OMA DRM 1.0 event handler, edit the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/EventService.properties` file and remove the initial pound sign (#) from the following statements:

```
eventservice.handler=OMARightsDelivery
eventservice.handler.OMARightsDelivery.classname=
com.sun.content.server.drm.oma.eventservice.OMADRM10RightsDeliveryHandler
eventservice.handler.OMARightsDelivery.events=content_purchased
```

CHAPTER 14

Submission Verifier Workflows

Content submitted to Sun Java System Content Delivery Server goes through a validation process that is managed by the submission verifier workflows. A workflow typically includes steps to validate the content using content validation adapters. The workflow that is executed is determined by the criteria that you specify. Content that does not require special processing must be processed by the default workflow.

This chapter includes the following topics:

- Content Validation Adapters
- Workflows Provided
- Creating a Workflow
- Specifying Workflow Criteria

14.1 Content Validation Adapters

A content validation adapter is used in a submission verifier workflow to process the content that is submitted to Content Delivery Server. Any preprocessing that is required before the content is accepted can be handled by an adapter. For example, adapters can be used to verify that the content meets the guidelines established by your enterprise, add code for digital rights management (DRM), or obfuscate the code.

14.1.1 Adapters Provided

Each step in a workflow must include the name of the content validation adapter to be run for that step. The following table describes the adapters that are provided with Content Delivery Server.

TABLE 14-1 Content Validation Adapters

Adapter	Description
`AddCapabilityAdapter`	Adds an additional device capability to the content that can be used to match the content to devices.
`AddDerivedEditionAdapter`	Stores the edition that is published, stocked, and downloaded.
`APIFilterAdapter`	Verifies that only APIs allowed by the developer plan assigned to the developer who submitted the content are used. It also determines which devices support the APIs that the content uses.
`AutoPublishAdapter`	Publishes content automatically without requiring action from the Catalog Manager administrator based on rules defined in the `$CDS_HOME/deployment/`*deployment-name*`/conf/AutoPublishRules.properties` file. See Section 1.7, "Setting Up Automatic Publishing" on page 1-9 for information on setting up the rules.
`CopyrightAdapter`	Ensures that copyrighted content is not stored locally.
`DRMAdapter`	Instruments a MIDlet with the DRM agent.
`ExternalToInternalAdapter`	Converts externally hosted content that was submitted using the Content Aggregator Interface to content that is stored locally.
`IAppliValidationAdapter`	Validates that the byte stream is an iAppli application archive file.
`MethodRedirectionAdapter`	Redirects certain method calls to methods that provide the special processing required by MIDlets instrumented with DRM agents.
`MIDletValidationAdapter`	Validates that the byte stream is a MIDlet application archive file.
`MIDletSigningAdapter`	Signs the MIDlet.
`MIDletPermissionsAdapter`	Adds permissions to the `MIDlet-Permissions` and `MIDlet-Permissions-Opt` attributes that are needed to run MIDlets that are instrumented with connected DRM agents.
`ProcessOmaDrmMessageAdapter`	Accepts content packaged in a DRM message as defined by OMA DRM 1.0 and adds the MIME type of the content as a required device capability.
`SetEditionWeightAdapter`	Specifies an edition weight for the content so when multiple editions match the capabilities of a device, the edition with the higher weight is delivered.

14.1.2 Writing an Adapter

If none of the adapters provided meets your needs, you can create your own adapter using the Content Validation API. See the *Sun Java System Content Delivery Server Customization Guide* for information on the Content Validation API.

If the adapter that you write needs values that cannot be known at the time the adapter is written, create a property file for the adapter. For example, if the adapter needs to know the location of a utility that it uses, create a property file that contains a property for the location. Set the location property to point to the directory that contains the utility on the system on which the adapter runs. Write the adapter to reference the location property when the location of the utility is needed. The property file that you create must be placed in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory.

14.1.3 Registering a Content Validation Adapter

To register an adapter that you wrote, add a statement in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/SubmissionVerifierAdapters.xml` file. If the adapter requires values in a property file, specify the name of the file in the `property-file` attribute. The property file must be in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory.

The following code example shows the registration statements for a sample adapter named `MyValidationAdapter` that requires a property file named `MyValidation.properties`.

CODE EXAMPLE 14-1 Sample Adapter Registration File

```
<adapter id="MyValidationAdapter" name="sample.package.MyValidationAdapter"
    propertyfile="MyValidation.properties"/>
```

14.2 Workflows Provided

Workflows provided with Content Delivery Server are defined in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/SubmissionVerifierWorkflows.xml` file. Predefined workflows are available for the following types of content:

- Java applications
- iAppli applications that use the DoJa library
- Copyrighted externally hosted content
- All other content (default workflow)

You can use these workflows as provided, or modify them as needed. For example, if you do not want to perform an action provided in a workflow, such as API filtering, comment out the step for that action. To add steps to a workflow or create a new workflow, see Section 14.3, "Creating a Workflow" on page 14-7. If you want to execute a workflow for content other than that specified by the default criteria, change the criteria as described in Section 14.4, "Specifying Workflow Criteria" on page 14-11.

Note – The `SubmissionVerifierWorkflows.xml` file must be saved as UTF-8. Make sure that the editor that you use supports this format.

14.2.1 Workflow for Java Applications

Content Delivery Server supports Java applications that use the MIDP 1.0 or MIDP 2.0 library. A default workflow is provided for use with most Java applications. A workflow for signing Java applications not protected by a CDS DRM agent is also provided.

14.2.1.1 Default Workflow for Java Applications

The default workflow for Java applications performs API filtering, stores the content, and checks if the content is to be published automatically. If no DRM method is used to protect the content, the content is stored in its original form. If a CDS DRM agent is used to protect the content, the workflow instruments the MIDlet. For MIDP 2.0 MIDlets, the workflow also adds any needed permissions to the `MIDlet-Permissions` and `MIDlet-Permissions-Opt` attributes and signs the MIDlet.

Disable an action by commenting out the associated step. For example, if you do not want to perform API filtering, comment out step 2 of the workflow.

Do not comment out the step named `AddingDerivedEdition`. This step stores the version of the content that subscribers download and is required.

To sign applications, a keystore file that contains the private key and trusted certificate for your installation is required. Use the `keytool` utility provided with the JDK™ software to create this file. You must specify `RSA` for the `keyalg` parameter. See the JDK software documentation for information.

If a CDS DRM agent is used to protect the content, you must set the following properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/cdsdrmagent.properties` file:

- `MIDletSigning.MIDP20.enabled`. Set this property to `true` to sign MIDP 2.0 MIDlets. Set to `false` if you do not want to sign the MIDlets.
- `MIDletSigning.KeyStoreFilePath`. Set this property to the fully qualified path name and file name of the keystore file.
- `MIDletSigning.KeyStorePassword`. Set this property to the password used to access the keystore.
- `MIDletSigning.KeyAlias`. Set this property to the alias of the private key that you want to use to sign the MIDlet.
- `MIDletSigning.KeyPassword`. Set this property to the password associated with the private key named by the key alias.

14.2.1.2 Workflow for Signing Java Applications

If you do not use a CDS DRM agent to protect content and have a need to sign MIDP 2.0 MIDlets, an alternate version of the workflow for Java applications is provided. To use this workflow, you must create criteria as described in Section 14.4, "Specifying Workflow Criteria" on page 14-11 that identifies the content that you want processed by this wokflow. The criteria for this workflow must precede the criteria for the default workflow for Java applications in the file.

The workflow for signed applications requires a keystore file that contains the private key and trusted certificate for your installation. Use the `keytool` utility provided with the JDK software to create this file. You must specify `RSA` for the `keyalg` parameter. See the JDK software documentation for information.

Edit the Signed Java Application Workflow in the `SubmissionVerifierWorkflows.xml` file and provide the fully qualified path and file name of the keystore file as the value for the `MIDletSigning.KeyStoreFilePath` in step 3 of the workflow. You must also provide values for the `MIDletSigning.KeyStorePassword`, `MIDletSigning.KeyAlias`, and `MIDletSigning.KeyPassword` properties. See Section 14.2.1.1, "Default Workflow for Java Applications" on page 14-4 for information on these properties.

14.2.2 Workflow for iAppli Applications

Content Delivery Server supports iAppli applications that use the DoJa library. The default workflow for iAppli applications performs API filtering, stores the content in its original form, and checks if the content is to be published automatically.

Disable an action by commenting out the associated step. Do not comment out the step named `AddingDerivedEdition`. This step stores the version of the content that subscribers download and is required.

14.2.3 Workflow for Externally Hosted Copyrighted Content

The workflow for externally hosted copyrighted content identifies content that must not be cached by Content Delivery Server because of copyright restrictions. This workflow creates an entry in the catalog for the content, ensures that only the metadata is stored for the content, and checks if the content is to be published automatically.

Disable an action by commenting out the associated step. The first two steps in this workflow are required. Do not comment out the step named `AddingDerivedEdition` or the step named `PreventingCopies`.

If you want Content Delivery Server to accept this type of content, remove the beginning and ending comment statements from the criteria named `isCopyrighted` that specifies execution of the workflow named `Copyrighted External Content Workflow`.

Any content except for Java applications and iAppli applications can be marked as copyrighted. To specify the content for which this workflow is executed, set the criteria as described in Section 14.4, “Specifying Workflow Criteria” on page 14-11.

14.2.4 Default Workflow

The default workflow is executed for all content that doesn’t match the criteria for any other workflow. The default workflow stores the content in its original form and checks if the content is to be published automatically.

Do not comment out the step named `AddingDerivedEdition`. This step stores the version of the content that subscribers download and is required.

14.3 Creating a Workflow

A submission verifier workflow describes the steps taken to validate and protect content submitted to Content Delivery Server. The following code example shows the workflow for externally hosted copyrighted content.

CODE EXAMPLE 14-2 Workflow for Externally Hosted Copyrighted Content

```
<workflow id="4" name="Copyrighted External Content Workflow"
          activation="manual">
  <desc>
    This workflow is used to ensure copyrighted external content is not stored
  </desc>
  <step-list>
    <step id="1" name="AddingDerivedEdition" adapter="AddDerivedEditionAdapter">
      <desc>
        This step adds a downloadable edition derived from the original
      </desc>
      <argument-list>
        <argument name="AddDerivedEdition.EditionNameSuffix"
                  kind="indirect" value="editionnamesuffix""/>
        <argument name="AddDerivedEdition.StoreOriginalBytes"
                  kind="direct" value="true"/>
      </argument-list>
    </step>
    <step id="2" name="PreventingCopies" adapter="CopyrightAdapter">
      <desc>
        This step removes any locally stored copies of the content
      </desc>
    </step>
    <step id="3" name="AutoPublish" adapter="AutoPublishAdapter"/>
  </step-list>
</workflow>
```

Each workflow that you define requires the following items:

- A unique workflow ID.
- A list of steps. Within the list, each step must have a unique ID and name and specify the name of the adapter to be executed.
- A step that executes `AddDerivedEditionAdapter`. This step stores the version of the content that a subscriber downloads. Every workflow that executes this adapter must contain at least one step. If your workflow modifies the content to create the edition that is downloaded, `AddDerivedEditionAdapter` must be

executed after the steps that perform the modifications and the argument `AddDerivedEdition.StoreOriginalBytes` must be set to `false`. Otherwise, the modifications are lost.

If your workflow creates more than one edition, the edition delivered to the subscriber depends on the capabilities of the device. If more than one edition matches the device, the last edition created that matches is the one delivered. For example, if steps 2, 5, and 7 in your workflow create unique editions of the content and the device is capable of running the editions created in steps 2 and 7, the edition created in step 7 is delivered.

- An argument list. If the adapter used in a step requires arguments, the step must include an argument list.

To automatically publish content based on rules defined in the `$CDS_HOME/deployment/`*deployment-name*`/conf/AutoPublishRules.properties` file, include a step that executes `AutoPublishAdapter`. See Section 1.7, "Setting Up Automatic Publishing" on page 1-9 for information on setting up the rules.

Add your workflow to the `$CDS_HOME/deployment/`*deployment-name*`/conf/SubmissionVerifierWorkflows.xml` file. See Section 14.4, "Specifying Workflow Criteria" on page 14-11 to specify the criteria for which your workflow is executed.

14.3.1 Using the Add Capability Adapter

The adapter `AddCapabilityAdapter` can be used to add required device capabilities to content during the verification process. For example, you can use this adapter to add the bit rate that a device is capable of handling for MP3 files.

The following code sample shows a workflow step that uses this adapter.

CODE EXAMPLE 14-3 Sample Workflow Step Using `AddCapabilityAdapter`

```
<step id="100" name="Cap" adapter="AddCapabilityAdapter">
  <desc>
    Add audio/mp3-24kbps to ccppaccept capability when
      the edition name contains "-24kbps"
  </desc>
  <argument-list>
    <argument name="AddCapability.EditionNameRegEx" kind="direct"
        value="-24kbps"/>
    <argument name="AddCapability.CapabilityName" kind="direct"
        value="ccppaccept"/>
    <argument name="AddCapability.CapabilityValue" kind="direct"
        value="audio/mp3-24kbps"/>
   </argument-list>
</step>
```

This step must appear before the AddDerivedEditionAdapter step so that the values from this step can be added to the derived edition. Otherwise, the values are never saved.

14.3.2 Using the External to Internal Adapter

The adapter ExternalToInternalAdapter can be used to convert externally hosted content that was submitted through the Content Aggregator Interface to local content.

The following code sample shows a workflow step that uses this adapter.

CODE EXAMPLE 14-4 Sample Workflow Step Using ExternalToInternalAdapter

```
<step id="300" name="EtoI" adapter="ExternalToInternalAdapter">
  <desc>
    Convert external content to internal content when
    the edition name contains "-extint"
  </desc>
  <argument-list>
    <argument name="ExternalToInternal.EditionNameRegEx" kind="direct"
      value="-extint"/>
  </argument-list>
</step>
```

This step must appear before the AddDerivedEditionAdapter step so that the conversion can be propagated to the derived edition. Otherwise, the values are never saved.

14.3.3 Using the Process OMA DRM Message Adapter

The adapter ProcessOmaDrmMessageAdapter can be used to accept content that is prewrapped with OMA DRM 1.0 protection and packaged in a DRM message. This adapter adds the MIME type of the protected content as a required device capability.

Note – Content that is prewrapped with OMA DRM protection must not have additional protection applied. If you are submitting prewrapped content, make sure that the DRM method assigned to the content type by the Catalog Manager administrator is None.

The following code sample shows a workflow step and criteria that use this adapter.

CODE EXAMPLE 14-5 Sample Workflow Step and Criteria Using `ProcessOmaDrmMessageAdapter`

```
<workflow id="5" name="OMA DRM Message Parser" activation="manual">
  <desc>Parse submitted DRM Message and update appropriate content
    capabilities depending on plain content wrapped inside DRM message</desc>
  <step-list>
    <step id="1" name="ProcessOMADRMMessage"
      adapter="ProcessOmaDrmMessageAdapter">
          <desc>This step parses the DRM wrapped content</desc>
    </step>
    <step id="2" name="AddingDerivedEdition"
        adapter="AddDerivedEditionAdapter">
    <desc>This step adds a downloadable edition derived from the original</desc>
      <argument-list>
        <argument name="AddDerivedEdition.EditionNameSuffix"
            kind="indirect" value="editionnamesuffix"/>
        <argument name="AddDerivedEdition.StoreOriginalBytes" kind="direct"
           value="true"/>
        </argument-list>
    </step>
  </step-list>
</workflow>

<criteria id="5" name="isDrmMessage">
  <desc>A sample validation workflow for a submitted OMA DRM Message</desc>
  <workflow-list>
    <workflow id="4">
      <argument-list>
       <argument name="editionnamesuffix" value="_OMADrmWrappedDownloadable"/>
      </argument-list>
    </workflow>
  </workflow-list>
  <criterion name="mime-type" value="application/vnd.oma.drm.message"/>
</criteria>
```

14.3.4 Using the Set Edition Weight Adapter

The adapter `SetEditionWeightAdapter` can be used to specify the edition weight for an edition of content. Edition weight can be used during capability matching to determine which edition is delivered to a device that matches multiple editions of content. If multiple editions match a device, the edition with the highest edition weight is delivered.

The following code sample shows a workflow step that uses this adapter.

CODE EXAMPLE 14-6 Sample Workflow Step Using `SetEditionWeightAdapter`

```
<step id="200" name="Weight" adapter="SetEditionWeightAdapter">
  <desc>
    Set the edition weight to 100 when
    the edition name contains "-24kbps"
  </desc>
  <argument-list>
    <argument name="SetEditionWeight.EditionNameRegEx" kind="direct"
      value="-24kbps"/>
    <argument name="SetEditionWeight.WeightValue" kind="direct" value="100"/>
  </argument-list>
</step>
```

This step must appear before the `AddDerivedEditionAdapter` step so that the values from this step can be added to the derived edition. Otherwise, the values are never saved.

14.4 Specifying Workflow Criteria

Each workflow must have at least one set of criteria that identifies the content for which the workflow is executed. This criteria is defined with the workflows in the `$CDS_HOME/deployment/`*deployment-name*`/conf/SubmissionVerifierWorkflows.xml` file.

Only one workflow is executed for each item of content. The workflow executed is determined by the first set of criteria that the content matches, so the order of criteria is important. If more than one criterion is specified in a set of criteria, all criterion must be met for the content to be considered a match.

CODE EXAMPLE 14-7 shows sample criteria for the workflow for externally hosted copyrighted content.

CODE EXAMPLE 14-7 Criteria List for the Copyrighted External Content Workflow

```
<criteria id="3" name="isCopyrighted">
  <desc>
    Sample criteria for copyrighted external content.
    Note: only external content can be copyrighted.
    Note: the only DRM supported for copyrighted content is None.
    Note: criterion names and values are case insensitive.
    Note: for location-type, specify e for external or i for
     internal.
  </desc>
```

CODE EXAMPLE 14-7 Criteria List for the Copyrighted External Content Workflow

```
  <workflow-list>
    <workflow id="4">
      <argument-list>
        <argument name=”editionnamesuffix value=”_Copyrighted”/>
      </argument-list>
    </workflow>
  </workflow-list>
  <criterion name="location-type" value="e"/>
  <criterion name="content-type" value="ringtone"/>
  <criterion name="mime-type" value="audio/mp3"/>
  <criterion name="developer-plan" value="copyrightplan"/>
  <criterion name="developer" value="composer"/>
</criteria>
```

Edit the criterion for an existing set of criteria or create additional sets of criteria to identify the content that you want to be processed by a workflow. For each new set of criteria:

- Provide a unique value for the `criteria id` attribute.
- For the `workflow id` attribute, specify the ID of the workflow that you want to assign. For example, for the copyrighted external content workflow, this value is `4`.
- If the specified workflow requires input values, include an argument list within the `workflow` element and provide the argument names and values.
- Order the sets of criteria within the file with the most restrictive set first. As soon as the content matches a set of criteria, no other criteria are checked. The last set of criteria must specify the default workflow and must not contain any criteria.

Use any combination of the following attributes as the criterion in a set of criteria. The workflow is executed if content matches all items specified.

- `location-type`. Specify `e` for externally hosted content, `i` for locally hosted content.
- `content-type`. Specify a content type that is supported by Content Delivery Server.
- `mime-type`. Specify a MIME type that is supported by Content Delivery Server.
- `developer-plan`. Specify the name of a Developer Plan.
- `developer`. Specify the user name of a developer.
- `MicroEdition-Configuration`. Specify the MicroEdition Configuration version for the MIDlet, for example, CLDC-1.0.
- `MicroEdition-Profile`. Specify the MicroEdition Profile version for the MIDlet, for example, MIDP-1.0.
- `ConfigurationVer`. Specify the Configuration Version for the iAppli application, for example, CLDC-1.0.

- `ProfileVer`. Specify the Profile Version for the iAppli application, for example, DOJA-1.0.

CHAPTER 15

Previews and Watermarks

Previews enable subscribers to sample content before purchasing. Image previews can be watermarked when stocked by the Vending Manager administrator, when viewed by the subscriber, or both when stocked and viewed.

Before submitting preview files, configure Content Delivery Server to handle previews and watermarks as needed.

The following topics are included in this chapter:

- Configuring Previews
- Configuring Watermarking

15.1 Configuring Previews

Optional captions can be used to identify the previews available. You can define a default caption for use when no caption is provided. You can also define the MIME types that you will use for audio previews files.

15.1.1 Setting the Default Caption

A preview caption is optional. If no caption is provided by the content provider or the administrator, a default caption is used. To set the default caption, set the `desktop.preview.untitled` property in the `$CDS_HOME/deployment/`*deployment-name*`/conf/SubscriberPortalLocaleResource.properties` file to the string of your choice. If you do not want a default caption, set this property to the empty string. If you have locale-specific resource files, set the property in the files for the locales that you use.

15.1.2 Identifying Audio Preview Files

To ensure that Content Delivery Server correctly recognizes files as audio previews, edit the `PlayableMimeTypes.config` file in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf` directory. Enter one or more regular expressions as defined by the `java.util.regex` package to identify the MIME types used in your system for audio files. Enter one expression per line, for example:

```
audio/.*
sound/.*
application/x-ert
```

15.2 Configuring Watermarking

Watermarking is the process of modifying a file to mark it as sample content. In Content Delivery Server, a watermark can be applied when content is stocked and when content is previewed by a subscriber. Watermarks are cumulative. If a watermark is applied when content is stocked and another watermark is applied when a preview is accessed by a subscriber, the preview contains both watermarks.

The Catalog Manager administrator works with the original preview file. If Content Delivery Server is configured to watermark content when content is stocked, the watermarked version of the preview file is stored in the Vending Manager and is available to the Vending Manager administrator. If Content Delivery Server is configured to watermark content when a preview is accessed, only the subscriber sees the watermarked version.

Watermarking is not enabled by default. To enable watermarking, follow the instructions in the following sections. The watermarking utility provided with the Content Deliver Server can watermark only files with the following MIME types:

- `image/png`
- `image/jpg`
- `image/gif`
- `image/tiff`

The default watermark is the string `Sample` in the center of the image as shown in the following figure.

FIGURE 15-1 Default Watermark

Currently, watermarks are supported for images only. Animated images cannot be watermarked. Externally hosted preview files that are copyrighted can be watermarked only when content is previewed, not when content is stocked. If the file is not copyrighted and the system is configured to watermark files when content is stocked, a copy of the externally hosted file is watermarked and stored in Content Delivery Server.

15.2.1 Installing the Java Advanced Imaging Image I/O Tools

To use the watermarking utility provided with Content Delivery Server, Java Advanced Imaging Image I/O Tools version 1.1 must be installed on the server on which the Vending Manager is deployed. These tools provide plug-ins to the Java Image I/O Framework for reading, writing, and streaming image formats and are available for multiple platforms. To install these tool, follow these steps:

1. **Identify the platform on which the Vending Manager is running.**

   Run the command `uname -a` on the server on which the Vending Manager is deployed. Use the results of this command to determine which binary build file you need.

2. **Download the appropriate binary build file for your platform.**

   Using the information from Step 1, download the release build file for version 1.1 from `https://jai-imageio.dev.java.net/binary-builds.html`. Download the file for your platform that ends with `-jdk`. For example, if the command returns the string `sparc`, download the `jai_imageio-1_1-lib-solaris-sparc-jdk.bin` file.

3. **Install the tools in your JDK software.**

   Follow the installation instructions at `http://download.java.net/media/jai-imageio/builds/release/1.1/INSTALL-jai_imageio.html` to install the file that you downloaded into your instance of the JDK software.

15.2.2 Watermarking When Content is Stocked

To enable watermarking when content is stocked, set the `watermarking.enabled.content-types` property in the `$CDS_HOME/deployment/`*deployment-name*`/conf/StockingWatermarking.properties` file to the content types to which watermarks are applied. Specify only content types defined to Content Delivery Server. The default is `image`. Currently, watermarks are supported for images only.

To use the default watermarking implementation for image content, do not change the other properties in this file.

You can use text, an image, or both text and an image as your watermark. Set the properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/StockingWatermarkingImage.properties` file to define the watermark to use. These properties are described in the following table.

TABLE 15-1 Properties for Defining a Watermark

Property	Description
`watermark.image.txt.source`	Text that is overlaid on the original image as a watermark. Maximum length is 255 characters. If no value is specified, all other properties beginning with `watermark.image.txt` are ignored. The initial value is `Sample`. **Note -** Make sure that the font specified for `watermark.image.txt.font.name` supports the characters used for the text watermark.
`watermark.image.txt.font.name`	Font used for the text watermark. To specify multiple fonts, separate the font name with a comma. If the system does not support any of the fonts specified, a substitute font is used. If no value is specified, `Helvetica,New Times,Courier,Arial` is used. The initial value is `New Times`.

TABLE 15-1 Properties for Defining a Watermark *(Continued)*

Property	Description
watermark.image.txt.position	Position and repetition factor used to place the watermark. The following values are valid: • TOP. Text is written across the top of the image. • BOTTOM. Text is written across the bottom of the image. • LEFT. Text is rotated and written along the left edge of the image. • RIGHT. Text is rotated and written along the right edge of the image. • TOP_BORDER. Text is written above the image. • BOTTOM_BORDER. Text is written below the image. • LEFT_BORDER. Text is rotated and written to the left of the image. • RIGHT_BORDER. Text is rotated and written to the right of the image. • DIAGONAL. Text is written diagonally across the image from the lower left corner to the upper right corner. • DIAGONAL_REPEATED. Text is written diagonally across the image in three places. • CENTER. Text is written horizontally across the middle of the image. • CENTER_REPEATED. Text is written horizontally across the image in three places. • NONE. Text is written in the upper left corner. If no value is specified, NONE is used. The initial value is CENTER.
watermark.image.txt.font.size	Size in points of the font used for the text watermark. Valid values are decimal numbers from 8 to 96 and FIT_IMAGE. Use FIT_IMAGE to have the size automatically adjusted to fit the original image. If no value is specified, FIT_IMAGE is used. The initial value is FIT_IMAGE.

TABLE 15-1 Properties for Defining a Watermark *(Continued)*

Property	Description
watermark.image.txt.font.colour	Color of the text watermark. The following values are valid: • Red, Green, Blue (RGB) values from 0 to 255, for example, 255,0,0 for red • WHITE • RED • BLUE • GREEN • BLACK • YELLOW • BROWN • PURPLE • PINK • ORANGE If no value is specified, BLACK is used. The initial value is BLACK.
watermark.image.txt.font.transparency	Transparency factor of the text watermark. Valid values are 0.0 through 1.0, where 0.0 is completely transparent and 1.0 is completely opaque. If no value is specified, 1.0 is used. The initial value is 0.75.
watermark.image.txt.font.modifier	Font modifier for the text watermark. Valid values are BOLD, ITALIC, UNDERLINE, and PLAIN. To specify more than one modifier, separate the values with a comma. If PLAIN is used, do not add any other modifier. If no value is specified, PLAIN is used. The initial value is PLAIN.
watermark.image.border.width	The width of the border in pixels that is added to all sides of the original image. A border can be used with both a text watermark and an image watermark. Specify any positive integer or 0 for no border. If no value is specified, 0 is used. The initial value is 0.
watermark.image.border.colour	The color of the border that is added to the original image. A border can be used with both a text watermark and an image watermark. Valid values are the same as the valid values for watermark.image.txt.font.colour. If no value is specified, BLACK is used. The initial value is an empty string.

TABLE 15-1 Properties for Defining a Watermark *(Continued)*

Property	Description
`watermark.image.img.source`	File name or URL for the image used as the watermark. The target of the URL must not be on a secured site and must be accessible through any firewall that exists between Content Delivery Server and the target. The file must be of one of the following MIME types: • `image/png` • `image/jpg` • `image/gif` • `image/tif` If no value is specified, all other properties beginning with `watermark.image.img` are ignored. The initial value is an empty string.
`watermark.image.img.position`	Position and repetition factor used to place the watermark. The following values are valid: • `TOP`. Watermark image is placed at the top of the image. • `BOTTOM`. Watermark image is placed at the bottom of the image. • `LEFT`. Watermark image is placed along the left edge of the image. • `RIGHT`. Watermark image is placed along the right edge of the image. • `TOP_BORDER`. Watermark image is placed above the image. • `BOTTOM_BORDER`. Watermark image is placed below the image. • `LEFT_BORDER`. Watermark image is placed to the left of the image. • `RIGHT_BORDER`. Watermark image is placed to the right of the image. • `DIAGONAL`. Watermark image is placed diagonally across the image from the lower left corner to the upper right corner. • `DIAGONAL_REPEATED`. Watermark image is placed diagonally across the image in three places. • `CENTER`. Watermark image is placed in the middle of the image. • `CENTER_REPEATED`. Watermark image is placed horizontally across the image in three places. • `NONE`. Watermark image is placed in the upper left corner. If no value is specified, `NONE` is used. The initial value is an empty string.

TABLE 15-1 Properties for Defining a Watermark *(Continued)*

Property	Description
`watermark.image.img.dimensions`	The X and Y dimensions in pixels of the watermark image, for example `10,10`. The watermark image is scaled if the dimensions exceed the actual size. If no value is specified, no overlay is applied. The initial value is an empty string.
`watermark.image.img.transparency`	Transparency factor of the watermark image. Valid values are `0.0` through `1.0`, where `0.0` is completely transparent and `1.0` is completely opaque. If no value is specified, `1.0` is used. The initial value is an empty string.

The server does not need to be restarted if you make changes to either of the property files. If you change the properties in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/StockingWatermarkingImage.properties` file, preview files for content stocked after the change use the updated watermark. Preview files for content stocked before the change retain the original watermark until one of the following actions occurs:

- Content is unstocked and then stocked again.
- Content is unpublished and then published again.
- An edition is updated or added to a content item.
- A preview set is changed by adding a file, deleting a file, changing a caption, or reordering the files.
- A custom preview set is created for an edition.

When one of the listed actions occurs for a content item, all preview files associated with that item are recreated using the original file and the updated watermark.

If the preview file is hosted externally, a watermarked copy of the file is stored in Content Delivery Server. If the externally hosted preview file is updated, a copy of the updated file is watermarked and stored in Content Delivery Server the next time a subscriber purchases or downloads an edition with which the preview file is associated.

15.2.3 Watermarking When Content is Previewed

To enable watermarking when content is previewed by the subscriber, set the `watermarking.enabled.content-types` property in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/DiscoveryWatermarking.properties` file to the content types to which watermarks are applied. The default is an empty string, so no watermark is applied when content is previewed.

To use the default watermarking implementation for image content, do not change the other properties in this file.

You can use text, an image, or both text and an image as a watermark. Set the properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/DiscoveryWatermarkingImage.properties` to define the watermark to use. These properties are described in TABLE 15-1. If watermarking is also enabled when content is stocked, the preview contains both watermarks.

The server does not need to be restarted if you make changes to either of the property files. If you change the properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/DiscoveryWatermarkingImage.properties` file, the next time a subscriber previews content the updated watermark is used.

Content Delivery Server provides an implementation of the Content Management API that handles watermarking as described in this section. To customize the watermarking process for your enterprise, you need to create your own implementation of the Content Management API. See the *Sun Java System Content Delivery Server Customization Guide* for information on the Content Management API.

CHAPTER 16

Content and Submission Formats

To accept submissions of iAppli applications, you must submit the required library to Content Delivery Server. To accept content submissions in a PAR file, you must configure Content Delivery Server for this option.

This chapter includes the following topics:

- Configure iAppli Support
- Configure PAR File Support

16.1 Configure iAppli Support

To support iAppli applications, add an entry in the database for the DoJa library and submit the library to Content Delivery Server.

16.1.1 Add the DoJa Library to the Database

An entry for the DoJa library is added to the Catalog Manager schema by default when the schema is created unless you removed the `iappli.sql` file from the `$CDS_HOME/dist/cds/database/cs` directory at the time you created the Catalog Manager schema. If you included iAppli support when you created the Catalog Manager schema, you just need to submit the DoJa library as described in Section 16.1.2, "Submit the DoJa Library" on page 16-2.

If you did not include iAppli support when you created the Catalog Manager schema, perform the following steps on the host where the Catalog Manager is deployed before submitting the library:

1. **Remove all files with the extension** `.sql` **from the** `$CDS_HOME/dist/cds/database/cs` **directory.**

   Save these files in another location.

2. **Copy the** `iappli.sql` **file into the** `$CDS_HOME/dist/cds/database/cs` **directory.**

3. **Run the following command:**

   `cdsi db data [-conf` *db-configuration-file*`] -cs`

   db-configuration-file is the name of the database configuration file that contains the information for creating the schemas. If *db-configuration-file* is not provided, the value specified for the `DEFAULT_DB` variable in the `init_env.sh` script is used. The switch `-cs` indicates that only the schema for the Catalog Manager is created.

4. **Restart Content Delivery Server.**

To verify that an entry exists in the database, log in to the Catalog Manager Administration console and display the list of device libraries to see the entry for the DoJa 1.0 library.

16.1.2 Submit the DoJa Library

Submit libraries through the Catalog Manager administration console. See the online help for additional information.

Make sure an entry for the DoJa library exists in the Catalog Manager database. If not, see the instructions in Section 16.1.1, "Add the DoJa Library to the Database" on page 16-1 for creating the entry. To submit the DoJa library, follow these steps:

1. **Start the Catalog Manager administration console from a browser window by typing the following address:**

   `http://`*hostname:port*`/admin/main`

   The Catalog Manager Log In page is displayed.

2. **Enter your administrator name and password.**

3. **Click Log In.**

   The Catalog Manager main page is displayed.

4. **Click Devices in the main menu bar.**

   The Device Management page is displayed.

5. **Click the Libraries tab.**

   The Device Libraries page is displayed.

6. **Click DOJA-1.0.**

   The Library Definition page is displayed.

7. **Click Upload JAR.**

   The library properties are displayed.

8. **For Select JAR File, enter the path of the library file or click Browse to locate the library file.**

   Do not change the name of the library.

9. **Click Next.**

   The items available for inclusion in the library are displayed.

10. **Select Global Package.**

11. **Click OK.**

    The library is added to the database and is available for use in Developer Plans to restrict the APIs used by developers.

12. **Click OK to close the confirmation page.**

Note – The library must be resubmitted whenever you reinstall your database.

16.2 Configure PAR File Support

Content Delivery Server accepts content in either Zip files or Provisioning Archive (PAR) files when submitted to Content Delivery Server. To support PAR files, you must set properties in the `$CDS_HOME/deployment/`*`deployment-name`*`/conf/DeveloperPortal.properties` file to map the PAR bundle types to Content Delivery Server content types.

Add a property for each type of content that you want to support using the following format:

`par.bundle-type.`*`type`*`=`*`content-type`*

type identifies the type of PAR bundle that is submitted. PAR bundle types are defined in the Java Specification Report (JSR) 124, which is available at http://www.jcp.org/aboutJava/communityprocess/final/jsr124/.

content-type identifies the type of content in the PAR bundle and must be one of the content types defined in Content Delivery Server. For example, the following statement shows the property for submitting image content in an image bundle:

```
par.bundle-type.image=image
```

You must restart Content Delivery Server after you make changes to the properties.

CHAPTER 17

Featured Content

Content Delivery Server uses custom fields and advanced search and sort capabilities to provide a way for the Vending Manager administrator to feature or promote content. Use this feature to specify the content that you want subscribers to see at the top of the list when they are browsing for content. You can configure featured content to be shown on only the PC-based Subscriber Portal, only the device-based Subscriber Portal, or on both portals.

The number of featured items shown is configurable. Content marked as featured is shown at the top of each page when browsing content and is also included in the standard list of content.

This chapter includes the following topics:

- Enabling Featured Content
- Configuring Featured Content

17.1 Enabling Featured Content

This feature is initially disabled for both the PC-based Subscriber Portal and the device-based Subscriber Portal. To support the ability to feature specific content items, follow these steps:

1. **Enable the feature in the portal of your choice.**

   Set the following properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/SubscriberPortal.properties` file:

   - `featured_content.desktop.show` - Set to `true` to show content marked as featured at the top of each page when browsing content in the PC-based Subscriber Portal. Set to `false` to not show an area for featured content.

- `featured_content.device.show` - Set to `true` to show content marked as featured at the top of each page when browsing content in the device-based Subscriber Portal. Set to `false` to not show an area for featured content.

2. **Set the number of featured items shown.**

   Set the following properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/SubscriberPortal.properties` file:

   - `featured_content.desktop.display.number` - Number of items to include in the list of featured content when shown on the PC-based Subscriber Portal. The default is `2`. This property is ignored if the `featured_content.desktop.show` property is set to `false`.
   - `featured_content.device.display.number` - Number of items to include in the list of featured content when shown on the device-based Subscriber Portal. The default is `1`. Keep in mind the limited display size of most devices when setting this property. This property is ignored if the `featured_content.device.show` property is set to `false`.

   If more content items than the default number are marked as featured, the items are shown in ascending order based on the value set for the feature order until the default number to display is reached.

3. **Define the custom fields that are needed.**

   The custom fields used by this feature are included in the `$CDS_HOME/deployment/`*deployment-name*`/conf/CustomField.properties` file as comments. Uncommment the properties shown in the following code sample by removing the pound sign (#) at the beginning of each line:

```
emf.all.isfeatured.scope=item
emf.all.isfeatured.required=false
emf.all.isfeatured.datatype=boolean
emf.all.isfeatured.editable=vm
emf.all.isfeatured.viewable=vm
emf.all.isfeatured.reserved=true

emf.all.featuredorder.scope=item
emf.all.featuredorder.required=false
emf.all.featuredorder.datatype=number
emf.all.featuredorder.editable=vm
emf.all.featuredorder.viewable=vm
emf.all.featuredorder.reserved=true
```

Note – The label properties for these fields are defined in the `$CDS_HOME/deployment/`*deployment-name*`/localization/VendingManagerMessages.properties` file. The properties are not initially commented out so no action is needed.

---

4. **Include the custom fields in the search schema.**

   The statements needed are included in the `$CDS_HOME/deployment/`*deployment_name*`/conf/vss/solr/conf/schema.xml` file as comments. Uncomment the following statements by removing the pound sign (#) from the beginning of the line:

```
<field name="emf.isfeatured" type="boolean" indexed="true" stored="true"/>
<field name="emf.featuredorder" type="sfloat" indexed="true" stored="true"/>
```

5. **Restart the Vending Manager.**

17.2 Configuring Featured Content

The Vending Manager administrator configures featured content using the Vending Manager administration console. The administrator identifies the content to feature and assigns the order in which the content appears. See the Vending Manager administration console online help for information on setting the custom fields associted with featured content.

PART IV User Interactions

This part of the *Sun Java System Content Delivery Server Integration and Configuration Guide* provides information on setting up the user interfaces and notification options provided by Content Delivery Server.

This part contains the following chapters:

- Portal Configuration
- Device-Specific User Interface Framework
- Notification Configuration

CHAPTER 18

Portal Configuration

The portals or interface components of Content Delivery Server are the Developer Portal, Subscriber Portal, Catalog Manager administration console, and the Vending Manager administration console. Content Delivery Server provides a number of properties that you can set to configure these interfaces to meet your needs. Each interface has its own file that contains the properties for that interface. A common file contains properties that apply to all interfaces.

Many configuration properties are described in other sections of this guide. This chapter provides information on some of the available properties that are not described in other sections.

This chapter includes the following topics:

- Setting the Common Properties
- Setting the Developer Portal Properties
- Setting the Subscriber Portal Properties
- Setting the Catalog Manager Properties
- Setting the Vending Manager Properties

18.1 Setting the Common Properties

The following table describes properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/CommonConsole.properties` file that you can modify to meet the needs of your system. These properties apply to all of the Content Delivery Server interfaces.

TABLE 18-1 Properties in the `CommonConsole.properties` File

Property	Description
`catalog.categories.max`	Maximum number of levels in the category hierarchy for both the Catalog Manager and the Vending Manager. For example, a value of `3` indicates that the main category can have subcategories and the subcategories can have subcategories. The default is `10`.
`handset.default.maximum.application.size`	Default size in bytes that is used as the maximum application size when a device is defined without specifying a value for this device attribute. The default is `64000`.
`handset.default.maximum.content.size`	Default size in bytes that is used as the maximum size for static content when a device is defined without specifying a value for this device attribute. The default is `100000`.
`web.common.upload_max_size`	Size in megabytes of the largest file that Content Delivery Server accepts. This size applies to all files submitted to Content Delivery Server, including submission packages and libraries. The default is `10`.

18.2 Setting the Developer Portal Properties

The following table describes properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/DeveloperPortal.properties` file that you can modify to meet the needs of your system. These properties apply only to the Developer Portal.

TABLE 18-2 Properties in the `DeveloperPortal.properties` File

Property	Description
`developer.pagination.display.max`	Maximum number of items displayed per page in the Developer Portal. The default is `20`.
`developer.pagination.display.threshold`	Number of items that must appear on a page before the list navigation bar is repeated at the bottom of the page. If fewer items than the number specified are available, the list navigation bar appears only at the top of the page. The default is `15`.
`external.content.minimum_refresh_period`	Minimum amount of time in milliseconds that Content Delivery Server waits before rechecking external content for updates. The default is `3600000`.

18.3 Setting the Subscriber Portal Properties

The following table describes properties in the`$CDS_HOME/deployment/`*deployment-name*`/conf/SubscriberPortal.properties` file that you can modify to meet the needs of your system. These properties apply only to the Subscriber Portal.

TABLE 18-3 Properties in the `SubscriberPortal.properties` File

Property	Description
`autoCreate.newDevice`	Enables an unknown device used by a subscriber to be added automatically to the Content Delivery Server database with information from the HTTP headers. Set this property to `true` to allow unknown devices to be added to the database automatically when used by a subscriber. Set to `false` to prevent devices from being added automatically.
`autoCreate.newUser`	Enables subscriber accounts to be created automatically in the Content Delivery Server database when an unregistered subscriber accesses the Subscriber Portal from a device. Whether the new subscriber is added to your subscriber database is dependent on the settings for the `auto_provision.unknown.user` properties. Set this property to `true` to create subscriber accounts in the Content Delivery Server database. Set to `false` to prevent the creation of new subscriber accounts.
`autoLogin.mobileId.enable`	Enables device login based on URL parameters instead of a user name and password. Automatic login based on mobile ID depends on information in the headers passed by the WAP gateway. Set this property to `true` to enable automatic login based on mobile ID. Set to `false` to disable automatic login.
`autoLogin.subscriberId.enable`	Enables device login based on URL parameters instead of a user name and password. Automatic login based on Subscriber ID looks for the `subId` parameter in the URL. This option is appropriate primarily for demonstration systems. Set this property to `true` to enable automatic login based on subscriber ID. Set to `false` to disable automatic login.

TABLE 18-3 Properties in the `SubscriberPortal.properties` File *(Continued)*

Property	Description
`autoLogin.uniqueId.enable`	Enables device login based on URL parameters instead of a user name and password. Automatic login based on unique ID depends on information in the headers passed by the WAP gateway. Set this property to `true` to enable automatic login based on unique ID. Set to `false` to disable automatic login.
`autoLogin.username.enable`	Enables device login based on URL parameters instead of a user name and password. Automatic login based on user name looks for the user parameter in the URL. This option is appropriate primarily for demonstration systems. Set this property to `true` to enable automatic login based on user name. Set to `false` to disable automatic login.
`default.pushType`	Specifies the default type of push message. Specify the default type of push message to send. Valid values are `sms` and `wap`. The corresponding property, either `wap.push.enable` or `sms.push.enable`, must also be set to `true`.
`descriptor.contentType.encoding.`*device-id*	Specifies the type of encoding for content type that is supported by the device identified by *device-id* in each property. To specify the default type of encoding, use 0 as the device ID. Set this property to the default character set encoding string to be included in the Content-Type HTTP header when downloading the Java Application Descriptor (JAD) file, for example, `ISO-8859-1`. If the device cannot handle the inclusion of the character set in the Content-Type header, set the property to `none`. **Note -** To get the ID for a specific device, place your mouse over the model name in the Device Management page in the Catalog Manager administration console. The ID is contained in the command shown in the status area of your browser.

TABLE 18-3 Properties in the `SubscriberPortal.properties` File *(Continued)*

Property	Description
`descriptor.outputStream.encoding.`*device-id*	Specifies the type of encoding for the output stream that is supported by the device identified by *device-id* in each property. To specify the default type of encoding, use 0 as the device ID. Set this property to the type of encoding used for the output stream, for example, `ISO-8859-1`. **Note -** To get the ID for a specific device, place your mouse over the model name in the Device Management page in the Catalog Manager administration console. The ID is contained in the command shown in the status area of your browser.
`desktop.confirmPurchase`	Specifies if a confirmation page is shown when content is purchased from the PC-based Subscriber Portal. Set this property to `true` to show a confirmation page. Set to `false` not show a confirmation page.
`desktop.display.numberOfItemsInContentList`	Specifies the number of items displayed for a search or browse result. If more items are available than are shown, navigation options are provided to view the entire result set. The default is `20`.
`desktop.display.numberOfItemsInMyCampaigns`	Specifies the number of items displayed for the subscriber's My Campaigns list. The most recent items added are displayed. If more items are available than are shown, a link is provided to view all items. Set to `-1` to display all items. The default is `5`.
`desktop.display.numberOfItemsInMyGiftsReceived`	Specifies the number of items. displayed when subscribers view the gifts they have received. The default is `5`.
`desktop.display.numberOfItemsInMyGiftsSent`	Specifies the number of items. displayed when subscribers view the gifts they have sent. The default is `5`.
`desktop.display.numberOfItemsInMyPurchases`	Specifies the number of items displayed for the subscriber's My Downloads list. The most recent items downloaded are displayed. If more items are available than are shown, a link is provided to view all items. Set to `-1` to display all items. The default is `5`.
`desktop.display.numberOfItemsInMyWishlist`	Specifies the number of items displayed for the subscriber's My Wish List. The most recent items added are displayed. If more items are available than are shown, a link is provided to view all items. Set to `-1` to display all items. The default is `5`.

TABLE 18-3 Properties in the `SubscriberPortal.properties` File *(Continued)*

Property	Description
`desktop.signup.disable_link`	Prevents an unknown subscriber from accessing the signup page on the desktop portal and registering for services. Set this property to `true` to disable the link to the signup page. Set to `false` to enable the link.
`desktop.signup.hide_link`	Prevents an unknown subscriber from accessing the signup page on the desktop portal and registering for services. Set this property to `true` to hide the link to the signup page. Set to `false` to show the link.
`device.confirmPurchase`	Specifies if subscribers must agree to a defined set of terms and conditions before each purchase. Set this property to `true` include a "Terms and Conditions" page in the purchase process. Set to `false` to not include a "Terms and Conditions" page.
`device.display.no_or_items_per_page`	Specifies the number of items that are displayed in the content list on the device-based Subscriber Portal. The default is `5`.
`display.price.my_downloads`	Specifies if the current price or the purchase price is shown when a subscriber views the content details from the My Downloads list. Also specifies if the current configuration of a bundle is shown or if the configuration at the time of purchase is shown. Set this property to `retail` to show the current price and configuration of bundles. Set to `purchase` to show the price and bundle configuration at the time of purchase.
`display.price.navigation`	Specifies if the current price or the purchase price is shown when a subscriber views the content details from the Select Content page. Also specifies if the current configuration of a bundle is shown or if the configuration at the time of purchase is shown. Set this property to `retail` to show all subscribers the current price and the current configuration of bundles. Set to `purchase` to show the price and bundle configuration at the time the subscriber purchased the content or bundle.
`dynamic_banner.default.path`	Specifies the path of the dynamic HTML that is displayed on the desktop home page. Set this property to the location for your system.

TABLE 18-3 Properties in the `SubscriberPortal.properties` File *(Continued)*

Property	Description
`gifts_restricted_to_recipients_plan`	Determines if subscribers can give content to another subscriber if the recipient's subscriber plan does not allow the recipient access to that content. Set this property to `true` to restrict the giver to giving only content allowed by the recipient's subscriber plan. Set to `false` to allow the giver to give content that is not in the recipient's subscriber plan. The default is `false`.
`lb.cookiename`	Specifies the name of the load balancer cookie. If a load balancer is being used to route traffic to different servers, a key-value pair can be added to every request to ensure that a subscriber's session remains on a specific server. Set this property to the name of the key evaluated by the load balancer.
`lb.cookievalue`	Specifies the value of the load balancer cookie. If a load balancer is being used to route traffic to different servers, a key-value pair can be added to every request to ensure that a subscriber's session remains on a specific server. Set this property to the value associated with the key evaluated by the load balancer.
`max.device.message.length`	Specifies the maximum number of bytes included in a message pushed to a device. The default is `10000`.
`newDevice.notification.emailAddress`	Sends notification of a new device. A notification email can be sent when a new device is automatically created. See `autoCreate.newDevice` to specify that devices are to be created automatically. Set this property to the email address to which notifications of new devices are sent.
`password_reminder.emailenabled`	Specifies if subscribers are provided with the option of having the password reminder sent to their email. Set this property to `true` to provide subscribers with the option. Set to `false` to hide the option.
`password_reminder.firstnameenabled`	Specifies if the First Name field is visible on the password reminder screen. Set this property to `true` to make the field visible. Set to `false` to hide the field.
`password_reminder.lastnameenabled`	Specifies if the Last Name field is visible on the password reminder screen. Set this property to `true` to make the field visible. Set to `false` to hide the field.

TABLE 18-3 Properties in the `SubscriberPortal.properties` File *(Continued)*

Property	Description
`password_reminder.template.filename`	Specifies the name of the style sheet used to generate password reminder messages. The default is `text/password_reminder.xsl`.
`password_reminder.webalertenabled`	Specifies if subscribers are provided with the option of having the password reminder sent to their device as an alert. Set this property to `true` to provide subscribers with the option. Set to `false` to hide the option.
`quarantine.newDevice`	Specifies if a device created automatically is quarantined. See `autoCreate.newDevice` to specify that devices are to be created automatically. Because the information from the HTTP headers might be incomplete, an option is provided to quarantine the device until the administrator completes the definition. Set this property to `true` to quarantine a newly added device. Set to `false` to make the device immediately available. Detected devices might not be fully defined.
`quick.browse.enabled`	Enables faster page loading by checking the number of items in a category and number of times an item was downloaded only once per user session. Set this property to `false` to check counts with every request. Set this property to `true` to check the counts once per user session. The default is `true`.
`quick.purchase.enabled`	Enables a faster purchase process by validating the purchase using the data received when the subscriber logged in or retrieved content instead of updating the data before validation. Validation checks include verifying that the subscriber is not disabled or that the content is not inactive. Set this property to `false` to make all validation checks at purchase time. Set this property to `true` to use the data retrieved when the subscriber logged in or the last time the subscriber retrieved the content. The default is `true`.
`share_content.gifting.expirationDays`	Specifies the number of days. a gift can be redeemed before it expires. The default is 30 days.
`share_content.template.filename`	Specifies the name of a style sheet from which the gift message is generated. The default is `text/share_content.xsl`.

TABLE 18-3 Properties in the `SubscriberPortal.properties` File *(Continued)*

Property	Description
`signup.unknown.user`	Determines if a user who is not in the subscriber database can register with Content Delivery Server. No check is done inside the function to determine if the user being registered is known to Content Delivery Server. Set this property to `true` to enable unknown users to register. Set to `false` to prevent unknown users from registering. The default is `true`.
`signup.unknown.user.firstName`	Specifies the value used for a subscriber's first name if a value is not provided when registering an unknown user. The default is `Signed-up User`.
`signup.unknown.user.lastName`	Specifies the value used for a subscriber's last name if a value is not provided when registering an unknown user. The default is `Signed-up User`.
`signup.unknown.user.middleName`	Specifies the value used for a subscriber's middle name if a value is not provided when registering an unknown user. The default is `Signed-up User`.
`signup.unknown.user.enabled`	Specifies if the subscriber account created for an unknown user is enabled when it is created. Set this property to `true` to enable the account. Set to `false` to disable the account. The default is `true`.
`signup.unknown.user.email`	Specifies the email domain used for a subscriber's email address if a value is not provided when registering an unknown user. The user's phone number is added to the beginning of the string to form the complete address. The default is `@external.com`.
`sms.bundle.items.threshold`	Specifies the maximum number of items in a bundle that are pushed directly to a device at one time. If a bundle contains more items for SMS delivery than this property specifies, the subscriber is provided with a link to the list of items to be downloaded individually. For example, if this property is set to 4 and the subscriber purchases a bundle that contains four or fewer items, all items are sent directly to the subscriber's device. If the bundle purchased contains more than four items, a link to the list of items is sent to the subscriber.
`sms.push.enable`	Specifies if SMS Push messages can be sent. Set this property to `true` to enable the sending of SMS Push messages. Set to `false` to disable SMS Push messages.

TABLE 18-3 Properties in the `SubscriberPortal.properties` File *(Continued)*

Property	Description
`subscriber.every_download.update_license`	Indicates if the license for content that a subscriber downloaded is updated when the license is changed. This property applies only to licenses that specify that the subscriber is charged on every download. Set this property to `true` to indicate that the subscriber is charged according to the new billing model when the billing model is changed. Set to `false` to indicate that the subscriber is charged according to the original billing model.
`user.security.max_login_attempts`	Specifies the number of times a subscriber can attempt to log in and fail before the account is locked. Set to `0` to allow unlimited attempts. The default is `3`.
`wap.push.enable`	Specifies if WAP Push messages can be sent. Set this property to `true` to enable the sending of WAP Push messages. Set to `false` to disable WAP Push messages.
`wml.display.no_of_items_per_page`	Specifies the maximum number of items. displayed per page by the device-based Subscriber Portal. The default is `5`.

18.4 Setting the Catalog Manager Properties

The following table describes properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/AdminConsole.properties` file that you can modify to meet the needs of your system. These properties apply only to the Catalog Manager administration console.

TABLE 18-4 Properties in the `AdminConsole.properties` File

Property	Description
`admin.pagination.display.max`	Specifies the maximum number of items displayed per page by the Catalog Manager administration console. The default is `20`.
`admin.pagination.display.threshold`	Specifies the number of items that must appear on a page before the list navigation bar is repeated at the bottom of the page. If fewer items than the number specified are available, the list navigation bar appears only at the top of the page. The default is `15`.
`preload_device_capabilities_on_device_list`	Determines if device capabilities for all devices are loaded the first time that the list of devices is accessed. Depending on the number of devices and the amount of content in your system, loading the capabilities for all devices might take several minutes or more and impact the performance of device-related activities such as viewing and editing devices. Set this property to `true` to load all device capabilities on first access. Set to `false` to load the device capabilities for a device only when that device is selected. The default is `true`. If you have many devices defined and notice a performance degradation when first accessing devices after a server restart, set this property to `false`.

18.5 Setting the Vending Manager Properties

The following table describes properties in the `$CDS_HOME/deployment/`*deployment-name*`/conf/VSAdminConsole.properties` file that you can set to meet the needs of your system. These properties apply only to the Vending Manager administration console.

TABLE 18-5 Properties in the `VSAdminConsole.properties` File

Property	Description
`admin.pagination.display.max`	Specifies the maximum number of items displayed per page in the Vending Manager administration console. The default is `20`.
`admin.pagination.display.threshold`	Specifies the number of items that must appear on a page before the list navigation bar is repeated at the bottom of the page. If fewer items than the number specified are available, the list navigation bar appears only at the top of the page. The default is `15`.

CHAPTER 19

Device-Specific User Interface Framework

The Subscriber Portal component of Content Delivery Server is a browser-based application that can be accessed on a PC or on the subscriber's device. Because the browsers used by different devices have different capabilities, Content Delivery Server provides a framework for generating Subscriber Portal pages that are tailored to the capabilities of the different devices. This framework applies only to the version of the Subscriber Portal that runs on a subscriber's device, not to the version that runs on a PC.

The version of the Subscriber Portal that runs on mobile devices was created using Apache Struts (see `http://jakarta.apache.org/struts/` for information on this framework). Subscriber Portal pages are provided for devices that use browsers based on WML and XHTML. The pages provided are suitable for many devices. However, if you are supporting a device that does not correctly show the pages of the Subscriber Portal or you want to take advantage of a device's special capabilities, you can create a version of the Subscriber Portal pages specifically for that device.

This chapter includes the following topics:

- Overview of the Framework
- Generating Pages for a Specific Device
- Modifying Pages for All Devices
- Adding a Custom Page

19.1 Overview of the Framework

The Subscriber Portal consists of pages created using JavaServer Pages™ technology (JSP™ pages). These JSP pages are generated from XML files that describe the pages to be produced and XSL style sheets that describe how the page elements are to be

rendered. A version of the Subscriber Portal, that is, one set of JSP pages, is generated for each style sheet. Each set of pages is stored in a subdirectory with the same name as the name of the style sheet. These subdirectories are in the following locations:

- `$CDS_HOME/deployment/`*deployment-name*`/sun/domains/`*server-domain*`/applications/j2ee-modules/CDSSubscriberPortal/device` if you are using Sun Java System Application Server.
- `$CDS_HOME/deployment/`*deployment-name*`/weblogic/domains/`*server-domain*`/applications/subscriber/device` if you are using WebLogic Server.

server-domain is the value specified for the `app.server.domain` property in the configuration file.

When the Catalog Manager administrator adds a device to the list of supported devices, one of the capabilities specified for the device is the browser type. The browser type specified for the device determines which version of the pages is used.

19.1.1 Page Definitions

The Struts framework is used to build the device-based Subscriber Portal. In the following cases, Content Delivery Server extends the Struts tag libraries to ensure that characters are XML or WML encoded:

- The Bean tag library is extended to contain the following items:
  - `<bean:encodedmessage>` - used in place of `<bean:message>`
  - `<bean:encodedwrite>` - used in place of `<bean:write>`
- The HTML tag library is extended to contain `<html:encodederrors>`, which is used in place of `<html:errors>`.

The files that contain the page definitions for the device-based Subscriber Portal are identified in the following table. These files are in the `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/page-defs` directory. The use of these files is described in Section 19.1.3, "Processes and Page Usage" on page 19-11.

TABLE 19-1 XML Files for Subscriber Portal Pages

File Name	Description
`_campaign.xml`	Shows the details for an individual campaign.
`_catalog_menu.xml`	Shows lists of links and is used to show the lists of categories, content, promotions, search results, My Downloads, and My Wish List.
`_confirm_unsubscribe.xml`	Prompts the subscriber to confirm the request to unsubscribe from an item of content.

TABLE 19-1 XML Files for Subscriber Portal Pages *(Continued)*

File Name	Description
_detail.xml	Shows the details for an item of content.
_device_error_msg.xml	Shows an error message.
_device_unsupported.xml	Notifies the subscriber that the device being used is not supported.
_download.xml	Prompts the subscriber to download an item of content.
_enter_coupon.xml	Prompts the subscriber to provide the information needed to redeem a coupon.
_gift_cancel_confirm.xml	Prompts the subscriber to confirm the request to cancel a gift subscription.
_gift_cancel_success.xml	Confirms that the gift subscription is cancelled.
_gift_details.xml	Shows the details for a gift that was sent to a subscriber.
_locale_selection.xml	Prompts subscribers to select their language preference.
_login.xml	Prompts the subscriber to log in.
_login_disabled.xml	Prevents the subscriber from logging in, and is shown if the subscriber attempts to log in to an account that is disabled, or if the wrong password is provided three times in a row.
_main_menu.xml	Shows the main menu shown when the subscriber logs in.
_manage_category.xml	Enables subscribers to select the categories that they want to see.
_my_gifts_menu.xml	Shows the links for gifts a subscriber gave and gifts a subscriber received. The associated link appears only if a subscriber gave or received at least one gift.
_preview.xml	Shows the preview if it is an image or plays the preview if it is an audio file. For an audio file, the page shows a link for replaying the preview.
_preview_list.xml	Shows a list of previews if more than one is available. The previews are identified by either the caption supplied by the content provider or Catalog Administrator or by a default caption if no caption was supplied. **Note -** If the first of multiple preview files is an audio file, the _preview.xml page is used and only the first file is available to the subscriber.
_purchase_confirm.xml	Prompts the subscriber to confirm the request to purchase content.

TABLE 19-1 XML Files for Subscriber Portal Pages *(Continued)*

File Name	Description
_search.xml	Prompts the subscriber for search criteria.
_share_content.xml	Enables the subscriber to share an item of content with another subscriber.
_share_content_confirm.xml	Prompts the subscriber to confirm the request to share content.
_share_content_receive.xml	Shows the recipient of shared content the details of that content.
_share_content_success.xml	Confirms that a message has been sent with a link to shared content.
_sms_sent.xml	Notifies the subscriber that the requested content is sent in an SMS message.
_unsub_success.xml	Confirms that the subscriber no longer has a subscription for an item of content.
_user_admin_menu.xml	Provides the options for administering an account, such as setting the language and managing categories.

The pages for the device-based Subscriber Portal are defined once using XML. The XML files are then processed with each existing style sheet to generate the JSP pages for each version of the Subscriber Portal that is needed. The following table describes the elements that can be used in each page.

TABLE 19-2 Page Elements

Element	Description
break	Inserts a single line break in an item element of a list element, between link elements in a navbar element, or within a text element.
button	Provides a button for a form and is used to submit data. This element appears under a form element. A button can contain the following elements: • align - The horizontal alignment of the button, used only for generating WML cards. Valid values are left, center, or right. The default is center. • label - The string displayed on the button. This element generally contains a <jsp> tag that contains a <bean:encodedmessage> or <bean:encodedwrite> tag. • name - A string used to identify the button to the handler that processes the form.
divider	Adds a horizontal line to a page between elements under either the view element or the form element.

TABLE 19-2 Page Elements *(Continued)*

Element	Description
`field`	Provides a field on a form in which a user can enter information. A `field` can contain the following elements: • `name` - Name of the field. This name maps to the form bean. • `type` - Type of field. Valid values are `text`, `password`, `hidden`, or `select`. • `label` - Label that appears on the form. This element generally contains a `<jsp>` tag that contains a `<bean:encodedmessage>` or `<bean:encodedwrite>` tag and is valid only for fields of type `text`, `password`, or `select`. • `size` - Size of the field. This element is valid only for fields of type `text` or `password`. • `maxlength` - Maximum length of the data a user can enter. This element is valid only for fields of type `text` or `password`. • `value` - Default value displayed for the field. • `option` **or** `optionlist` - List of items for a field that is of type `select`. See the description for `option, optionlist` in this table. • `multiple` - Flag that indicates if multiple items can be selected. Set this element to `true` to allow multiple selections. Set to `false` to allow only single selection. • `format` - Input mask for the field, used only for generating WML cards. This element is valid only for fields of type `text` or `password`. See the WML specification for valid values. • `required` - Flag that indicates if a field is required, used only for generating WML cards. This element is valid only for fields of type text or password. Set this element to `true` to indicate that the field is required. Set to `false` to indicate that the field is not required. The default is `false`.
`form`	Describes a form for the page and appears under the `view` element. A `view` can have only one `form`. A `form` can contain the following elements: • `action` - URL to which the data is sent when the form is submitted. • `button` - Button on the form. A form can contain multiple buttons. See the description for `button` in this table. • `field` - Field on the form. A form can contain multiple fields. See the description for `field` in this table. • `divider` - See the description for `divider` in this table. • `image` - See the description for `image` in this table. • `jsp` - See the description for `jsp` in this table. • `list` - See the description for `list` in this table. • `navbar` - See the description for `navbar` in this table. • `text` - See the description for `text` in this table. Using the provided style sheets, `button` elements are displayed at the bottom of the form. All other elements are displayed in the order in which they appear.

TABLE 19-2 Page Elements *(Continued)*

Element	Description
image	Provides an image for a page and appears under a view element or a form element. An image can contain the following elements: • name - Name of the file that contains the image. The style sheet provides the path and the suffix for the image. The default path is http://*host:port*/subscriber/static/media/device and the default suffix is gif. For example, if the image is in /static/media/device/logo.gif, set name to logo. Do not include this element if the path element is included. • path - URL path to the file that contains the image, for example, http://server1.com/web/images/logo.gif. Do not include this element if the name element is included. • alt - Alternate text for an image. This element generally contains a <jsp> tag that contains a <bean:encodedmessage> or <bean:encodedwrite> tag.
jsp	Contains code that uses syntax for the JSP technology (JSP syntax). This element can appear under any element and does not contain any elements. Include the code in a CDATA section so that the XML page definitions remain valid. The code can include the following JSP syntax: • Directives, such as the include directive • Scripting elements and variables • Actions • Custom tags, such as the Logic, Bean, and HTML tags that are provided by Struts. For example, the following code segment shows the use of scripting elements and variables to set the value of the title element: <title> <jsp><!CDATA[<%=title%>]]></jsp> </title> To include the code at the top of the page, set the attribute header for this element to true. If the attribute is set to false or does not appear, the code in jsp elements is generated in the order in which the elements appear.

TABLE 19-2 Page Elements *(Continued)*

Element	Description

`link`	Provides a link on the page and appears under either the `navbar` element or an `item` element of a `list` element. A `link` contains the following elements:
	• `name` - Text displayed on the page. This element generally contains a `<jsp>` tag that contains a `<bean:encodedmessage>` or `<bean:encodedwrite>` tag.
	• `url` - URL to which the link points. According to the style sheet, the value is the parameter for a method in a Java class file and must include a `jsp` element that contains a valid `String` literal or `String` variable. See the example at the end of the description for this element.
	• `accesskey` - Flag that indicates whether to include a shortcut key to access the item. Set to `true` to include an access key. Set to `false` or omit the element to not include an access key. This attribute is ignored if a browser cannot handle access keys.
	For example, the following code segment shows a link to a page outside of the Subscriber Portal.
	`<link>`
	`  <name>Yahoo</name>`
	`  <url><jsp>"http://wap.yahoo.com"</jsp></url>`
	`</link>`
	To indicate that the link is used to download purchased content, set the attribute `download` for this element to `true`. This attribute is used by the style sheets to determine the tag to use for some devices and content types.

`list`	Provides a list of items for a page and appears under the `view` element or the `form` element. The `list` element contains one or more `item` elements. An `item` element contains a `link` element, a break element, or a `jsp` element that contains code that writes text, for example, `<bean:encodedmessage>` or `<bean:encodedwrite>`.
`navbar`	Provides a set of links that are displayed as a group, such as `ok` and `cancel` or `yes` and `no`. These links are automatically separated by a navspacer. The `navbar` element appears under the `view` element or the `form` element and contains one or more `link` elements and `navspacer` elements. To specify the orientation of the navigation bar, set the attribute `orientation`. Valid values are `horizontal` and `vertical`. The default is `vertical`. **Note -** Some browsers might not have the capability to group the links.
`navspacer`	Divides links in a navigation bar and appears between `link` elements under the `navbar` element. **Note -** Links in a `navbar` are automatically divided with `navspacers`. Use a `navspacer` element to further divide the links.

TABLE 19-2 Page Elements *(Continued)*

Element	Description
option, optionlist	Describes the list of items included in a field of type select and appears under the field element. Use option when the list of items is known. Use optionlist when the list is dynamically generated. An option contains the following elements: • name - Name of the item. • value - Value assigned to the item. According to the style sheet, the value of the optionlist element must be valid HTML statements that contain all of the option elements for a select statement. Use a jsp element for the code that generates the valid HTML option tags, for example: <optionlist> <jsp><![CDATA[<%=getOptions()%>]]></jsp> </optionlist> The method getOptions is expected to return a String that contains option elements similar to the following example: <option value="1">One</option> <option value="2">Two</option> <option value="3">Three</option>
text	Provides the text for the page and appears under the view element or the form element. The text element generally contains a <jsp> tag that contains a <bean:encodedmessage>, <bean:encodedwrite>, or <html:encodederrors> tag and can also contain one or more break or link elements. This element can have the following attributes: • alignment - Valid values are left, right, and center. • type - Valid values are error and bold.
title	Provides the title used for the page and appears under the view element. The title element generally contains a <jsp> tag that contains a <bean:encodedmessage> or <bean:encodedwrite> tag.
view	Describes the page and is the top-most element. This element contains the following elements: • divider • form • jsp • list • navbar • text • title For a WML-based browser, the view represents a card. For an HTML-based browser, the view represents the body. This element can have the following attributes: • main - Set to true to include header and footer images on a page. Set to false to not include header and footer images. • error - Set to true to identify the page as the error page. Only one page definition can have this attribute set to true.

The following code example shows a page that contains a form. This sample is the file `$CDS_HOME/dist/cds/markup_generation/page_defs/_login.xml`.

CODE EXAMPLE 19-1 Sample Page With Form

```
<?xml version="1.0"?>
<view>
  <title>
    <jsp><![CDATA[<bean:encodedmessage key="device.login.title"/>]]></jsp>
  </title>
  <jsp><![CDATA[<logic:messagesPresent>]]></jsp>
  <text type="error">
    <jsp><![CDATA[<html:encodederrors/>]]></jsp>
  </text>
  <jsp><![CDATA[</logic:messagesPresent>]]></jsp>
  <form>
    <action>
      <jsp>Web.getWeb().getActionURL2(SubscriberConstants.ACTION_DEVICE_LOGIN,
           (UrlParams)null, response)</jsp>
    </action>
    <field>
      <name>username</name>
      <type>text</type>
      <size>10</size>
      <maxlength>40</maxlength>
      <label>
        <jsp><![CDATA[<bean:encodedmessage key="device.login.username"/>]]></jsp>
      </label>
    </field>
    <field>
      <name>password</name>
      <type>password</type>
      <size>10</size>
      <maxlength>40</maxlength>
      <label>
        <jsp><![CDATA[<bean:encodedmessage key="device.login.password"/>]]></jsp>
      </label>
    </field>
    <button>
      <name>
        <jsp><![CDATA[<%=SubscriberConstants.BUTTON_SUBMIT%>]]></jsp>
      </name>
      <label>
        <jsp><![CDATA[<bean:encodedmessage key="device.login.loginLink"/>]]></jsp>
      </label>
    </button>
  </form>
</view>
```

The following code example shows a page that contains links to other pages. This sample is the file `$CDS_HOME/dist/cds/markup_generation/page_defs/_main_menu.xml`.

CODE EXAMPLE 19-2 Sample Page With Links

```
<?xml version="1.0"?>
<view main=”true”>
  <title>
    <jsp><![CDATA[<bean:encodedmessage key="device.menu.main.title"/>]]></jsp>
  </title>
  <list>
    <jsp><![CDATA[<logic:iterate id="element"
         name="<%=SubscriberConstants.ATTR_MENU_LIST%>"
         type="com.sun.content.server.subscriberportal.common.ListItem"
         indexId="index">]]></jsp>
    <item>
      <link>
        <name>
          <jsp><![CDATA[<bean:encodedwrite name="element"
               property="name"/>]]></jsp>
        </name>
        <accesskey>true</accesskey>
        <url>
          <jsp>element.getUrl()</jsp>
        </url>
      </link>
    </item>
    <jsp><![CDATA[</logic:iterate>]]></jsp>
  </list>
</view>
```

19.1.2 Style Sheets

Style sheets provide templates that describe how to render each element used to define a JSP page. These elements are described in TABLE 19-2. Style sheets interpret the XML page definitions and create the JSP pages for a given device or class of devices.

Set up style sheets to render markup according to the capabilities of the device used. For example, when an XML definition specifies a link, the style sheet for one type of browser might render the link in color, where the style sheet for a different type of browser might render the link with an underline.

Content Delivery Server provides the following style sheets with the product. These style sheets are in the `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/stylesheets` directory.

- `WML-1_1.xsl` - Provides basic functionality for devices that support WML 1.1.
- `WML-1_2.xsl` - Extends the style sheet for WML 1.1 to support WML 1.2 fuctionality, including access keys.
- `XHTML-Basic.xsl` - Provides basic functionality for devices that support XHTML.
- `XHTML-Color.xsl` - Extends the style sheet for XHTML to include support for header and footer images and Cascading Style Sheet (CSS) color schemes.
- `XHTML-AU.xsl` - Extends the XHTML-Color style sheet to support the AU-System browser.
- `XHTML-IAppli.xsl` - Extends the XHTML-Color style sheet to support an iAppli browser.
- `XHTML-NokiaSeries40.xsl` - Extends the XHTML-Color style sheet to support the browser on Nokia Series 40 devices.
- `XHTML-SE.xsl` - Extends the XHTML-Color style sheet to support the browser on newer Sony Ericsson devices.
- `XHTML-Alternate.xsl` - Extends the XHTML-Basic style sheet to support the browser on older devices or devices with less features.
- `XHTML-Symbian.xsl` - Extends the XHTML-Color style sheet to support the Symbian browser and larger images.
- `XHTML-UP.xsl` - Extends the XHTML-Color style sheet to support the Openwave UP browser.
- `XHTML-Motorola.xsl` - Extends the XHTML-Color style sheet to support the Mobile Internet Browser (MIB) 2.2 (or later) on newer Motorola devices.

These style sheets are suitable for many devices. However, if the pages of the Subscriber Portal do not display well on a device, a new style sheet can be created to define a different rendering of the elements. Only those elements that do not display well need to be included in the new style sheet. For example, if a device uses the `XHTML-Alternate.xsl` style sheet and only links and fields render poorly, create a style sheet that imports `XHTML-ALternate.xsl` and includes definitions for only links and fields.

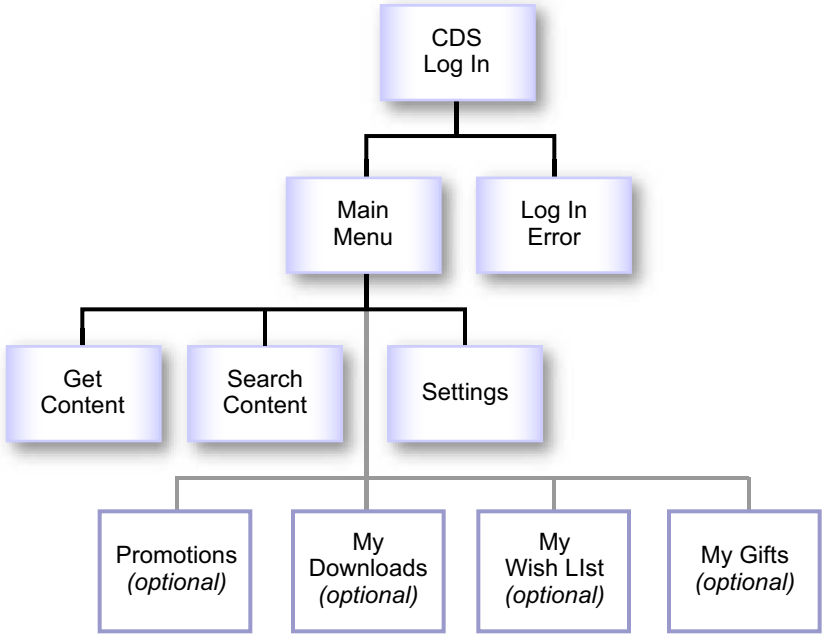
19.1.3 Processes and Page Usage

The following sections describe the general process flow of the primary functions of the Subscriber Portal. These descriptions identify the pages shown to subscribers on their device and the files used to generate the pages. The files are described in TABLE 19-1.

19.1.3.1 Log In Process

The following figure shows the process of logging in to the Subscriber Portal and the options available after the subscriber is logged in. The description that follows the figure identifies the XML file used for each page.

FIGURE 19-1 Log In Process



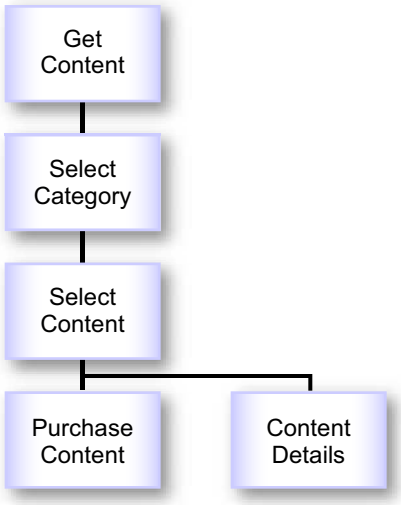
1. The CDS Log In page, generated from `_login.xml`, is the first page shown to the subscriber.
2. If login is successful, the main page of the Subscriber Portal, generated from `_main_menu.xml`, is shown. If the login fails, an error is shown on the CDS Log In page. If the subscriber's account is disabled, or the login fails three times in a row, a page in which the login is disabled, generated from `_login_disabled.xml`, is shown.
3. From the Main Menu page, the following actions are available:
   - View content, see Section 19.1.3.2, "View Content Process" on page 19-13.
   - Search for content, see Section 19.1.3.3, "Search for Content Process" on page 19-14.
   - Set preferences, see Section 19.1.3.4, "Set Preferences Process" on page 19-15.

- View promotions, if available, see Section 19.1.3.5, "View Promotions Process" on page 19-16.
- View My Downloads, if available, see Section 19.1.3.6, "View the My Downloads List Process" on page 19-16.
- View My Wish List, if available, see Section 19.1.3.7, "View My Wish List Process" on page 19-16.
- View My Gifts, if available, see Section 19.1.3.8, "View the My Gifts List Process" on page 19-17.

19.1.3.2 View Content Process

The following figure shows the process of viewing available content.

FIGURE 19-2 View Content Process



1. When the subscriber clicks Get Content on the Main Menu, the Select Category page, generated from `_catalog_menu.xml`, is shown. This page shows the list of categories that are available to the subscriber.
2. When the subscriber clicks a category name in the list of categories, the Select Content page, generated from `_catalog_menu.xml`, is shown. This page shows the list of content and content bundles that are available to the subscriber.

3. When the subscriber clicks an item of content in the content list, either the Purchase Content page or the Content Details page, both generated from `_detail.xml`, is shown.
   - The Purchase Content page is shown if the subscriber has not yet purchased the content. The following options are provided:
     - Purchase
     - Preview (if a preview is available)
     - Try (if a trial is available)
     - I Have a Coupon
     - Buy for a Friend
     - Tell a Friend
   - The Content Details page is shown if the subscriber has purchased the content. The content details and the following options are provided:
     - Go to Download
     - Preview (if a preview is available)
     - Unsubscribe (if the subscriber has an active subscription for the content)
     - I Have a Coupon
     - Buy for a Friend
     - Tell a Friend

   See Section 19.1.3.9, "Purchase or Download Process" on page 19-17 for a description of these options.

19.1.3.3 Search for Content Process

The following figure shows the process of searching content.

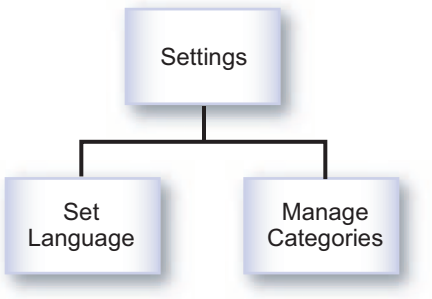
FIGURE 19-3 Search for Content Process

1. When the subscriber clicks Search Content on the Main Menu, the Search Content page, generated from `_search.xml`, is shown. This page prompts the subscriber for the search criteria.
2. After the search completes, the Search Results page, generated from `_catalog_menu.xml`, is shown. This page shows the list of content that matches the search criteria.

19.1.3.4 Set Preferences Process

The following figure shows the process of setting preferences.

FIGURE 19-4 Set Preferences Process



When the subscriber clicks Settings on the Main Menu, the Settings page, generated from `_user_admin_menu.xml`, is shown. This page provides the following options:

- **Set Language**. When this option is selected, the Set Language page, generated from `_locale_selection.xml`, is shown. This page enables the subscriber to select the language to be used.
- **Customize Categories**. When this option is selected, the Manage Categories page, generated from `_manage_category.xml`, is shown. This page enables the subscriber to choose the categories to be shown and the order in which the categories are shown.

19.1.3.5 View Promotions Process

The Promotions option is available on the Main Menu page only if campaigns are set up in the Vending Manager and the option to receive notifications about promotions is on in the subscriber's profile. When the subscriber clicks Promotions, the Promotions page, generated from `_catalog_menu.xml`, is shown. This page shows the list of promotions that are available.

When the subscriber selects a promotion, the details for that promotion, generated from `_campaign.xml`, are shown. When the subscriber clicks an item of content included in the promotion, the Purchase Content page, generated from `_detail.xml`, is shown. From this page, the subscriber can download the item or purchase the item for a friend. See Section 19.1.3.9, "Purchase or Download Process" on page 19-17 for information on the purchase process.

19.1.3.6 View the My Downloads List Process

The My Downloads list is available on the Main Menu page only if the subscriber previously purchased content. When the subscriber clicks My Downloads, the My Downloads page, generated from `_catalog_menu.xml`, is shown. This page shows the list of content that the subscriber has purchased.

When the subscriber selects an item of content, the Content Details page, generated from `_detail.xml`, is shown. From this page, the subscriber can download the item, purchase the item for a friend, or tell a friend about the item. See Section 19.1.3.9, "Purchase or Download Process" on page 19-17 for information on the purchase process.

19.1.3.7 View My Wish List Process

My Wish List is available on the Main Menu page only if the subscriber added content to the wish list when browsing content using the PC-based Subscriber Portal. When the subscriber clicks My Wish List, the My Wish List page, generated from `_catalog_menu.xml`, is shown. This page shows the list of content that the subscriber has downloaded.

When the subscriber selects an item of content, the Purchase Content page, generated from `_detail.xml`, is shown. See Section 19.1.3.9, "Purchase or Download Process" on page 19-17 for information on the purchase process.

19.1.3.8 View the My Gifts List Process

The My Gifts list is available on the Main Menu page only if the subscriber received a gift or purchased a gift for another subscriber. When the subscriber clicks My Gifts, the My Gifts page, generated from `_my_gifts_menu.xml`, is shown.

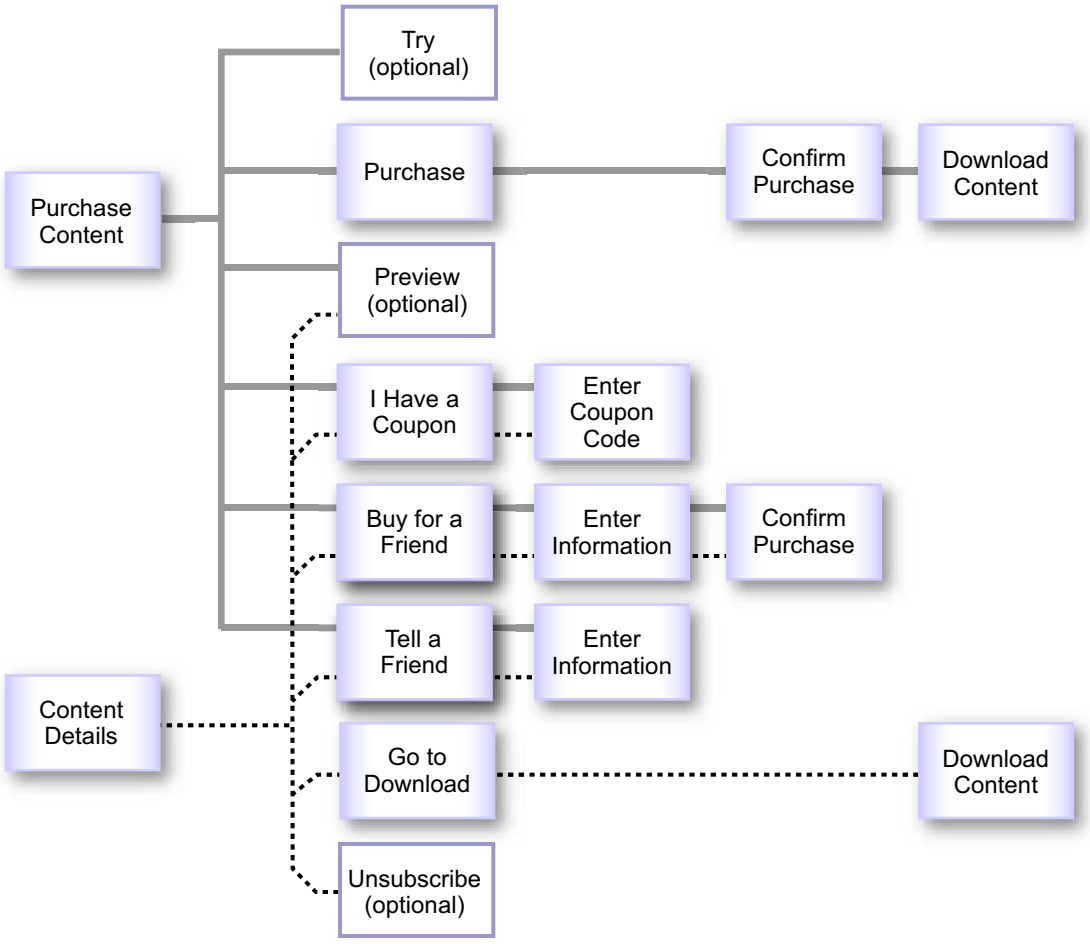
If the subscriber received a gift, the My Gifts page has a link for Gifts Received. When the subscriber clicks the Gifts Received link, the Gifts Received page, generated from `_catalog_menu.xml`, is shown. This page shows the list of content that the subscriber received as gifts. When the subscriber selects an item of content, the details for the gift, generated from `_gift_details.xml`, are shown. To accept a gift, the subscriber clicks the Download Gift link and the Download page, generated from `_download.xml`, is shown.

If the subscriber purchased a gift for another subscriber, the My Gifts page includes a link for Gifts Sent. When the subscriber clicks the Gifts Sent link, the Gifts Sent page, generated from `_catalog_menu.xml`, is shown. This page shows the list of content that the subscriber purchased as gifts. When the subscriber selects an item of content, the details for the gift, generated from `_gift_details.xml`, are shown. Gifts of content charged on a subscription basis have an option to cancel the gift. If the subscriber cancels a gift subscription, a page confirming the request, generated from `_gift_cancel_confirm.xml`, is shown. If the request completes successfully, a notification page, generated from `_gift_cancel_success.xml`, is shown.

19.1.3.9 Purchase or Download Process

The following figure shows the process of either purchasing content or downloading content already purchased. The purchase process is initiated from the Purchase Content page. The download process is initiated from the Content Details page. Both pages are generated from the `_detail.xml` file.

FIGURE 19-5 Purchase Process



The Purchase Content page is shown if the subscriber has not yet purchased the selected content or if the license requires that the content be purchased again. The Content Details page is shown if the subscriber has purchased the selected content. The following options are provided as noted:

- **Try**. Subscribers can try an item before purchasing it by clicking Try. The Download page, generated from `_download.xml`, is shown.

  The try option is available only on the Purchase Content page and only if the administrator set up a trial for the item.

- **Purchase**. Subscribers can purchase content for themselves by clicking Purchase. The Confirm Terms page, generated from `_purchase_confirm.xml`, is shown. If the subscriber confirms the purchase, the Download page, generated from

_download.xml, is shown. If the subscriber downloads content that is delivered in an SMS message, a confirmation page, generated from _sms_sent.xml, is shown.

The purchase option is available only on the Purchase Content page.

- **Preview**. Subscribers can preview the content if a preview is available. If only one preview is available or if the first preview is an audio file, the Preview page, generated from _preview.xml, is shown. For an image preview, this page shows the image. For an audio preview, this page plays the audio file and shows a link for replaying the preview. For both image and audio, this page has a link to either purchase or download the content, depending on whether the subscriber has previously purchased the item.

  If multiple previews are available, the list of previews, generated from _preview_list.xml, is shown. When a preview is selected, the Preview page is shown as described above.

  The preview option is available on both pages if the content can be previewed.

- **I Have a Coupon**. If subscribers have a coupon code, they can purchase content at a discount for themselves or as a gift by clicking I Have a Coupon. The Enter Coupon Code page, generated from _enter_coupon.xml, is shown. After a coupon code is entered, the Purchase Content page, generated from _detail.xml, is shown and the purchase process is started with the discounted price.

  The coupon option is available on both pages.

- **Buy for a Friend**. Subscribers can purchase content as a gift for another subscriber by clicking Buy for a Friend. The Buy for a Friend page, generated from _share_content.xml, is shown. After the information for the recipient is provided, a confirmation page, generated from _share_content_confirm.xml, is shown. If the request completes successfully, a notification page, generated from _share_content_success.xml, is shown.

  The message that the recipient of the gift receives includes a link to the gift. When the recipient clicks the link and accesses the Subscriber Portal, a notification about the gift, generated from _share_content_receive.xml, is shown.

  The gift option is available on both pages.

- **Tell a Friend**. Subscribers can notify another subscriber of content in which they might be interested by clicking Tell a Friend. The Tell a Friend page, generated from _share_content.xml, is shown. If the request completes successfully, a notification page, generated from _share_content_success.xml, is shown.

  The message sent to the friend includes a link to the content. When the recipient clicks the link and accesses the Subscriber Portal, a notification about the content, generated from _share_content_receive.xml, is shown.

  The share option is available on both pages.

- **Go to Download**. Subscribers can download content they purchased by clicking Go to Download. The Download page, generated from `_download.xml`, is shown. If the subscriber downloads content that is delivered in an SMS message, a confirmation page, generated from `_sms_sent.xml`, is shown.

  The download option is available only on the Content Details page.

- **Unsubscribe**. An option to unsubscribe is available if the subscriber has a subscription for the item. When the subscriber unsubscribes, a confirmation page, generated from `_confirm_unsubscribe.xml`, is shown. When the subscription is successfully cancelled, a notification page, generated from `_unsub_success.xml`, is shown.

  The unsubscribe option is available only on the Content Details page and only if the subscriber has a subscription to the content.

19.2 Generating Pages for a Specific Device

Generate the device-specific pages on a test system to avoid disrupting the production system. When you are sure that the generated pages are correct, move the files created to your production system.

To generate a version of the Subscriber Portal that is tailored to a specific device or set of devices, follow these steps:

1. **Create an XSL style sheet in the** `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/stylesheets` **directory.**

   Give the new style sheet a name that identifies the device or device category for which the generated pages will be used. Import a parent style sheet based on the browser category that the device uses. For example, if the browser category is XHTML, include the statement `<xsl:import href="XHTML-Basic.xsl"/>` in the style sheet. Follow the structure of the parent style sheet. Templates are needed only for those elements that do not render correctly, or to take advantage of advanced capabilities that are offered on a device.

2. **Run the command** `cdsi genmarkup -ss` *stylesheet* **where** *stylesheet* **is the name of the style sheet that you created in** Step 1 **and does not include the** `.xsl` **extension.**

   This command processes all of the XML files in the `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/page-defs` directory against the new style sheet and generates a set of JSP pages. The results are placed in the `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/jsps/`*stylesheet* directory.

3. **Move the directory that contains the generated JSP pages to the Subscriber Portal application directory on each application server where you deployed a Subscriber Portal.**

- For Sun Java System Application Server, `$CDS_HOME/deployment/`*deployment-name*`/sun/domains/`*server-domain*`/applications/j2ee-modules/CDSSubscriberPortal/device`
- For WebLogic Server, `$CDS_HOME/deployment/`*deployment-name*`/weblogic/domains/`*server-domain*`/applications/subscriberportal/device`

4. **Add the name of the style sheet to the `$CDS_HOME/deployment/`*deployment-name*`/conf/browser.config` file.**

   This file maintains the list of supported browser types from which the Catalog Manager administrator chooses when adding a device. For example, if you created a style sheet named `XHTML-newBrowser.xsl` to support a new browser type, add the following statement to the file:

   ```
   device.markup.browser.option=XHTML-newBrowser
   ```

19.3 Modifying Pages for All Devices

If you want to make changes to the version of the Subscriber Portal that is run on mobile devices, you can change the page definitions and regenerate the JSP pages. The changes that you make are only seen when the Subscriber Portal is accessed from a mobile device. Changes made to the XML page definitions do not affect the version of the Subscriber Portal that runs on a PC.

Make the changes on a test system to avoid disrupting the production system. When you are sure that the generated pages are correct, move the files created to your production system.

To change a page, follow these steps:

1. **Edit an existing XML file in the `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/page-defs` directory or create a new file and add it to the directory.**

   See Section 19.1.1, "Page Definitions" on page 19-2 for a description of the elements that can be included in a file.

2. **Run the command `cdsi genmarkup -ss all`.**

   This command processes all of the XML files in the `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/page-defs` directory against all of the style sheets in the `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/stylesheets` directory and generates a new set of JSP

pages for each style sheet. Each set of pages is placed in the `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/jsps/`*stylesheet* directory, where *stylesheet* is the name of the style sheet used to generate the pages.

3. **Move the directories that contain the generated JSP pages to the Subscriber Portal application directory on each application server where you deployed a Subscriber Portal.**

- For Sun Java System Application Server, `$CDS_HOME/deployment/`*deployment-name*`/sun/domains/`*server-domain*`/applications/j2ee-modules/CDSSubscriberPortal/device`
- For WebLogic Server, `$CDS_HOME/deployment/`*deployment-name*`/weblogic/domains/`*server-domain*`/applications/subscriberportal/device`

19.4 Adding a Custom Page

If you have a good understanding of Java technology, XML, and Struts, you can add a page to the Subscriber Portal to customize the page flow.

As an example of customizing the page flow, this section describes the process for adding a Terms and Conditions page before the main menu is shown. To add the page, follow these steps:

1. **In the `$CDS_HOME/deployment/`*deployment-name*`/markup_generation/page-defs` directory, create the XML file that defines the page that you want to add.**

   See Section 19.1.1, "Page Definitions" on page 19-2 for information on the page elements that you can use. The Terms and Conditions page for this example could be defined as shown in the following code example.

CODE EXAMPLE 19-3 Terms and Conditions Page Definition

```
<?xml version="1.0"?>
<!-- Copyright (c) 2003 Sun Microsystems, Inc. All rights reserved -->
<!-- SUN PROPRIETARY/CONFIDENTIAL. -->
<!-- Use is subject to license terms. -->
<view>
  <title>
    <jsp><![CDATA[<bean:encodedmessage key="device.newPage.title"/>]]></jsp>
  </title>
  <text>
    <jsp><![CDATA[<bean:encodedmessage
         key="device.newPage.instructions"/>]]></jsp>
  </text>
  <navbar orientation="horizontal">
```

CODE EXAMPLE 19-3 Terms and Conditions Page Definition *(Continued)*

```
    <link>
      <name>
        <jsp><![CDATA[<bean:encodedmessage
             key="device.newPage.continue"/>]]></jsp>
      </name>
      <url>
        <jsp>Web.getWeb().getActionURL("/device_direct_url.do",
             null,response)</jsp>
      </url>
    </link>
  </navbar>
</view>
```

2. **Add the strings used in the page definition to each language version of the** `SubscriberPortalLocaleResource.properties` **file in the** `$CDS_HOME/deployment/`*`deployment-name`*`/localization` **directory.**

   The name of the properties that you add corresponds to the values that you specified for the `key` parameters, for example:

```
device.newPage.title=Terms and Conditions
device.newPage.instructions=Here are the latest updates to the site's Terms and
Conditions
device.newPage.continue=Continue
```

3. **Generate all of the pages for the Subscriber Portal.**

   a. **Run the command** `cdsi genmarkup -ss all`**.**

      This command processes all of the XML files in the `$CDS_HOME/deployment/`*`deployment-name`*`/markup_generation/page-defs` directory against all of the style sheets in the `$CDS_HOME/deployment/`*`deployment-name`*`/markup_generation/stylesheets` directory and generates a new set of JSP pages for each style sheet. Each set of pages is placed in the `$CDS_HOME/deployment/`*`deployment-name`*`/markup_generation/jsps/`*`stylesheet`* directory, where *stylesheet* is the name of the style sheet used to generate the pages.

   b. **Move the directories that contain the generated JSP pages to the Subscriber Portal application directory on each application server where you deployed a Subscriber Portal.**

   - For Sun Java System Application Server, `$CDS_HOME/deployment/`*`deployment-name`*`/sun/domains/`*`server-domain`*`/applications/j2ee-modules/CDSSubscriberPortal/device`
   - For WebLogic Server, `$CDS_HOME/deployment/`*`deployment-name`*`/weblogic/domains/`*`server-domain`*`/applications/subscriberportal/device`

4. **Create a handler that contains the business logic for the new page.**

   The handler must extend `com.sun.content.server.subscriberportal.device.BaseDeviceHandler` and implement the `doExecute` method. CODE EXAMPLE 19-4 shows a call to this method. See the output of the Javadoc tool provided with Content Delivery Server for information on this class.

   Place the compiled class in a JAR file in one of the following locations:

- For Sun Java System Application Server, `$CDS_HOME/deployment/`*`deployment-name`*`/sun/domains/`*`server-domain`*`/applications/j2ee-modules/CDSSubscriberPortal/WEB-INF/lib`
- For WebLogic Server, `$CDS_HOME/deployment/`*`deployment-name`*`/weblogic/domains/`*`server-domain`*`/applications/subscriberportal/WEB-INF/lib`.

  The following code example shows a sample handler that determines if the Terms and Conditions page should be shown.

CODE EXAMPLE 19-4 Sample Handler

```
package com.sun.content.server.cdsexample;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import com.sun.content.server.subscriberapi.IApiContext;

/**
 * Show a new page if an external service determines that the user needs to
 * see this page.
 */
public class CheckNewPageHandler extends BaseDeviceHandler
{
    public ActionForward doExecute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception
    {
        IApiContext apiContext = getApiContext(request);

        // ExternalService class needs to be implemented
       boolean showPage = ExternalService.showPage(apiContext.getMobileId());

        if (showPage)
            return mapping.findForward("show_page");
        else
```

CODE EXAMPLE 19-4 Sample Handler *(Continued)*

```
            return mapping.findForward("do_not_show_page");
    }
}
```

5. **Edit the `struts-config.xml` file to indicate how to handle the page in the page flow.**

   This file is in one of the following locations:

   - For Sun Java System Application Server, `$CDS_HOME/deployment/`*deployment-name*`/sun/domains/`*server-domain*`/applications/j2ee-modules/CDSSubscriberPortal/WEB-INF`
   - For WebLogic Server, `$CDS_HOME/deployment/`*deployment-name*`/weblogic/domains/`*server-domain*`/applications/subscriberportal/WEB-INF`

   a. **Remove the following section of code:**

```
<action path="/device_provision"
  type="com.sun.content.server.subscriberportal.device.ProvisionUserHandler">
  <forward name="device_select_locale" path="/dv5.do"/>
  <forward name="success" path="/dv42.do"/>
</action>
```

   b. **Add the following code in place of the section that you removed:**

```
<action path="/device_provision"
  type="com.sun.content.server.subscriberportal.device.ProvisionUserHandler">
  <forward name="device_select_locale" path="/dv5.do"/>
  <forward name="success" path="/check_new_page.do"/>
</action>
<action path="/check_new_page"
  type=" com.sun.content.server.cdsexample.CheckNewPageHandler">
  <forward name="show_page" path="/device_show_new_page.do"/>
  <forward name="do_not_show_page " path="/device_direct_url.do"/>
</action>
<action path="/device_show_new_page"
  type="com.sun.content.server.subscriberportal.device.ReturnSuccessHandler">
  <forward name="success" path="/View?pg=_new_page.jsp"/>
</action>
```

6. **Restart the server.**

CHAPTER 20

Notification Configuration

Sun Java System Content Delivery Server provides notifications to content developers about content they submitted and to subscribers about updates and promotions. You can determine the types of notifications sent to content developers and the opt-in default for subscribers.

This chapter includes the following topics:

- Configuring Developer Notifications
- Configuring the Default for Subscriber Notifications

20.1 Configuring Developer Notifications

Developers submit content to Content Delivery Server through the Developer Portal. You can set up Content Delivery Server to send email notifications to developers whenever action is taken on the content that they submitted.

To set up notifications, follow these steps:

1. **Make sure that the** `default.external.developerportal.uri` **property in the** `$CDS_HOME/deployment/`*deployment-name*`/conf/CDS.properties` **file points to the location of the Developer Portal.**

   Use the format `http://`*hostname:port*`/developer/`, where *hostname:port* is the host name and port number of the host on which the Developer Portal is running.

2. **Set the following properties in the** `$CDS_HOME/deployment/`*deployment-name*`/conf/EventService.properties` **file.**

   - `eventservice.developer.email.enabled` - Status of email notifications. Set this property to `true` to automatically send notifications when action is taken on content. Set to `false` to disable the sending of notifications. The default is `false`.

- `eventservice.developer.email.from` - Address used as the address from which the notification is sent.
- `eventservice.developer.email.admin` - Address to which developers can respond if they have questions about the notification.
- `eventservice.developer.email.template.propertychanged` - Fully qualified location of the XSL file that contains the description of the notification sent when any content property is changed.
- `eventservice.developer.email.template.submitted` - Fully qualified location of the XSL file that contains the description of the notification sent when content is received or rejected by Content Delivery Server.
- `eventservice.developer.email.template.statuschanged` - Fully qualified location of the XSL file that contains the description of the notification sent when the status of content is changed.

3. **(Optional) Customize the notifications that are sent.**

   You can customize notifications by editing the default templates provided with Content Delivery Server or by creating your own XSL files.

   CODE EXAMPLE 20-1 provides the definition of the notification templates that you can follow. If you create your own files, you must update the properties described in the previous step that identify the location of the files.

CODE EXAMPLE 20-1 Definition of Notification Templates

```
/*
 * Definition of XML documents that are piped to the
 * XSL transformation sheets.
 *
 * <dn>
 *  <developer>Joe Developer</developer>
 *
 *  <submission> <!-- when content was submitted (even if failed) -->
 *    <succeeded> <!-- may be zero or more -->
 *      <name>Name used to submit</name>
 *      <id>ID used to submit</id>
 *      <url>URL this content is viewable at</url>
 *    </succeeded>
 *    <failed> <!-- may be zero or more -->
 *      <name>Name used to submit</name>
 *      <id>ID used to submit</id>
 *      <errormsg>Error message</errormsg>
 *      <errorlog>Lengthy error info</errorlog>
 *    </failed>
 *  <submission>
 *
 *  <statuschanged>
 *
 *    <!-- The possible status values are:
```

CODE EXAMPLE 20-1 Definition of Notification Templates *(Continued)*

```
 *
 *    pending
 *    denied
 *    published
 *    unpublished
 *    deleted
 *    new
 *
 *    -->
 *
 *    <newstatus>status</newstatus>
 *
 *    <name>Name used to submit</name>
 *    <id>content id</id>
 *    <url>URL where this content is accessible</url>
 *    <message>Message left for developer</message>
 *  </statuschanged>>
 *
 *  <propertychanged>
 *    <name>Name used to submit</name>
 *    <id>content id</id>
 *    <url>URL where this content is accessible</url>
 *    <textproperty> <!-- zero or more of those -->
 *      <name>property name</name>
 *      <oldvalue>old property value</oldvalue>
 *      <newvalue>new property value</newvalue>
 *    </textproperty>
 *
 *    <!-- binary property is special, since it doesn't have a
 *         displayable value, so only name is mentioned.
 *    -->
 *    <binaryproperty>property name</binaryproperty> <!-- zero or
 *           more -->
 *    <priceproperty> <!-- zero or one of this, only for suggested
 *           price -->
 *      <!-- note the pricedata tag. It is used to unify the
 *           structure to make XSLT transformation easier.
 *      -->
 *      <oldvalue><pricedata>
 *        <billing1>
 *          <!-- the following is $13.27, but helps localization
 *               if broken in such a way. There is also a "number"
 *               tag which gives the price in cents (coins)
 *           -->
 *          <price>
 *            <number>1327</number>
 *            <bills>13</bills>
```

CODE EXAMPLE 20-1 Definition of Notification Templates *(Continued)*

```
 *             <coins>27</coins>
 *           </price>
 *           <usage_count>usage_count</usage_count>
 *          <usage_period_days>usage_period_days</usage_period_days>
 *           <recurring>recurring</recurring>
 *         </billing1>
 *         <billing2>
 *           <price><bills>13</bills><coins>27</coins></price>
 *           <frequency>frequency</frequency>
 *           <recurring>recurring</recurring>
 *         </billing2>
 *         <billing3>
 *           <price><bills>13</bills><coins>27</coins></price>
 *           <usage_count>usage_count</usage_count>
 *         </billing3>
 *       </pricedata></oldvalue>
 *       <newvalue><!-- the same as for oldvalue --> </newvalue>
 *     </priceproperty>
 *
 *  </propertychanged>
 *
 *  <adminemail>Email address developer can inquire at</adminemail>
 * </dn>
 */
```

20.2 Configuring the Default for Subscriber Notifications

Content Delivery Server sends notifications to subscribers about updates or promotions. Subscribers can choose to receive these notifications by setting the option when they set their preferences for their account. You can set the default to either opt-in or opt-out. The initial default is opt-in.

To change the default option, set the `user.profile.optin` property in the `$CDS_HOME/deployment/`*deployment-name*`/conf/security.config` file. Set this property to `true` to set the default to opt-in. Set this property to `false` to set the default to opt-out.

Index

Symbols

_campaign.xml file, 19-2, 19-16
_catalog_menu.xml file, 19-2, 19-13, 19-15, 19-16, 19-17
_confirm_unsubscribe.xml file, 19-2, 19-20
_detail.xml file, 19-3, 19-14, 19-16, 19-17, 19-19
_device_error_msg.xml file, 19-3
_device_unsupported.xml file, 19-3
_download.xml file, 19-3, 19-17, 19-18, 19-19, 19-20
_enter_coupon.xml file, 19-3, 19-19
_gift_cancel_confirm.xml file, 19-3, 19-17
_gift_cancel_success.xml file, 19-3, 19-17
_gift_details.xml file, 19-3, 19-17
_locale_selection.xml file, 19-3, 19-16
_login.xml file, 19-3, 19-12
_login_disabled.xml file, 19-3, 19-12
_main_menu.xml file, 19-3, 19-12
_manage_category.xml file, 19-3, 19-16
_my_gifts_menu.xml file, 19-3, 19-17
_preview.xml file, 19-3, 19-19
_preview_list.xml file, 19-3, 19-19
_purchase_confirm.xml file, 19-3, 19-18
_search.xml file, 19-4, 19-15
_share_content.xml file, 19-4, 19-19
_share_content_confirm.xml file, 19-4, 19-19
_share_content_receive.xml file, 19-4, 19-19
_share_content_success.xml file, 19-4, 19-19
_sms_sent.xml file, 19-4, 19-19, 19-20
_unsub_success.xml file, 19-4, 19-20
_user_admin_menu.xml file, 19-4, 19-15

A

access permission, 3-8
adapter
  push listener
    overview, 9-1
    register, 9-6
  push sender
    CIMD2 properties, 8-5
    overview, 8-1
    register, 8-6
    SMS HTTP properties, 8-4
  streaming, 5-2
  subscriber, LDAP, 3-1
  WAP gateway
    default, 7-2
    Nokia Activ Server 2.0.1, 7-2
    Nokia Artus WAP gateway, 7-2
    Openwave WAP gateway, 7-2
AddCapabilityAdapter adapter, 14-2, 14-8
AddDerivedEditionAdapter adapter, 14-2, 14-7
AdminConsole.properties file, 18-12
alarms, 1-3
API
  Event Service, 2-1
  Messaging, 8-1, 9-1
  User Profile, 3-1
  WAP Gateway, 7-1
API filtering, 14-4
APIFilterAdapter adapter, 14-2
audio preview, 15-2

authorization key, 1-17
auto_provision.unknown.user property, 3-11
auto_provision.unknown.user.email property, 3-12
auto_provision.unknown.user.enabled property, 3-12
auto_provision.unknown.user.firstName property, 3-12
auto_provision.unknown.user.lastName property, 3-12
auto_provision.unknown.user.middleName property, 3-12
auto_provision.unknown.userpeer property, 3-11
auto_provision.unknown.userpeer.giftin g property, 3-11
auto_provision.unknown.userpeer.mobile _originate property, 3-11
auto-login user, 18-5
automatic provisioning
  unknown user, 3-11
  unknown user peer, 1-18
automatic publishing, 1-9
AutoPublishRules.properties file, 1-9, 14-8

B

BaseDeviceHandler class, 19-24
billing adapter, 2-1
Billing API, 2-1
billing information, 2-5
billing integration, 2-1
billing records, 2-3
binary content, push, 8-6
browse, quick, 18-9
browser.config file, 19-21
bundle configuration shown, 18-7
buy for a friend, 19-19

C

Catalog Manager properties, 18-12
category depth, 18-2
CDS.properties file, 1-9, 12-1, 20-1
cdsdrmagent.properties file, 13-4, 14-5
cdsi command, 19-20, 19-21, 19-23
CDSSnmp.properties file, 1-2
CIMD2
  SMS push listener, 9-5
  SMS push sender, 8-5
CIMD2 properties, 9-5
Combined Delivery, 13-5
CommonConsole.properties file, 18-2
conf.xml file, 3-2
confirm purchase
  desktop, 18-6
  device, 18-7
confirmation message, 10-2
ConfirmListener.properties file, 10-2
CopyrightAdapter adapter, 14-2
coupon, 19-19
csv_record.xsl, 2-4
currency symbol, 12-1
custom fields
  defining, 1-11
  featured content, 17-1
  hints, 1-13
  labels, 1-13
  properties, 1-11
  searching, 11-9
CustomField.properties file, 1-11, 17-2
customized reports, 1-5

D

database synchronization, 1-4
default_record.xsl, 2-4
desktop content list, items in, 18-6
developer notifications, 20-1
Developer Portal properties, 18-3
DeveloperPortal.properties file, 18-3
device
  notification, 18-8
  preload capabilities, 18-12
  quarantine, 18-9
Device Client web services, 1-17
device-specific pages
  generating, 19-20
  modifying, 19-21
  page elements, 19-4
device-specific user interface framework, 19-1
digital rights management *See* DRM
digital rights object, 13-5

DiscoveryWatermarking.properties file, 15-8
doExecute method, 19-24
DoJa library, submitting, 16-2
DRM
  agents, 13-3
  configuration, 13-1
  event handler, 13-7
  methods
    enabling and disabling, 13-2
    supported, 13-1
  non-compliant devices, 13-6
  preferred delivery type, 13-5
DRM Fusion Toolkit, 4-1
DRMAdapter adapter, 14-2
DRMAgent property, 13-4
dynamic banner, 18-7

E

encoding, content type, 18-5
Event Service API, 2-1
Event Service, configuring, 1-3
EventService.properties file, 1-3, 13-7, 20-1
expiration, share content, 18-9
expired item, deactivate, 1-17
external content ID, 2-8
external content, refresh period, 18-3
ExternalToInternalAdapter adapter, 14-2, 14-9

F

featured content, 17-1
field names
  Catalog Manager, 11-3
  Developer Portal, 11-3
  Subscriber Portal, 11-5
  Vending Manager, 11-5
files, maximum size, 18-2
Forward Lock, 13-5
framework, user interface, 19-1
FulfillmentService.properties file, 1-18

G

genmarkup command, 19-20, 19-21, 19-23
gift, 19-19
gift restriction, 18-8
grace period, 12-3

H

hints, custom fields, 1-13

I

iAppli applications, 14-5, 16-1
iappli.sql file, 16-1
IAppliValidationAdapter adapter, 14-2
items in
  content list, device, 18-7
  My Campaigns, 18-6
  My Downloads, 18-6
  My Gifts received, 18-6
  My Gifts sent, 18-6
  My WIsh List, 18-6
  SMS bundle, 18-10
items per page
  Catalog Manager, 18-12
  Developer Portal, 18-3
  Subscriber Portal, 18-11
  Vending Manager, 18-13

J

Java Advanced Imaging Image I/O Tools, 15-3

L

labels, custom fields, 1-13
LDAP
  access permission, 3-8
  mapping file for, 3-2
  sample mapping file, 3-6
  tuning, 3-8
LDAP subscriber adapter, 3-1
load balancer cookie, 18-8
login attempts, 18-11

M

mail service, configuring, 10-1
mapping file, LDAP, 3-2, 3-6
maximum size
  application, 18-2
  content, 18-2
  files, 18-2
message length, 18-8

Messaging API, 8-1, 9-1
Messaging Service, 10-1
MethodRedirectionAdapter adapter, 14-2
MIDletPermissionsAdapter adapter, 14-2
MIDletSigningAdapter adapter, 14-2
MIDletValidationAdapter adapter, 14-2
mobile originated push message, 9-2
Monitoring Service, configuring, 1-1
MsgService.properties file, 8-5, 10-1
My Campaigns, items in, 18-6
My Downloads
  items in, 18-6
  process, 19-16
My Gifts, 19-17
My Gifts received, items in, 18-6
My Gifts sent, items in, 18-6
My Wish List
  items in, 18-6
  process, 19-16

N

Nokia Activ Server 2.0.1, 7-2
Nokia Atrus WAP gateway, 7-2
notification templates, 20-2
notification, device, 18-8
notifications
  developer, 20-1
  subscriber, opt-in, 20-4
number of login attempts, 18-11

O

oma.drm10.plain property, 13-6
oma.drm10.rights property, 13-5
omadrm10.properties file, 13-5, 13-6
Openwave WAP gateway, 7-2

P

page elements, 19-4
page threshold
  Catalog Manager, 18-12
  Developer Portal, 18-3
  Vending Manager, 18-13
pages, Subscriber Portal, 19-2
PAR file, 16-3
password reminder, 18-8
permission, access, 3-8
PlayableMimeTypes.config file, 15-2
popularity, 1-14
  customizing definition, 1-15
  default definition, 1-14
  recalculating, 1-16
port number, Confirm Listener Service, 10-2
Postpaid Service, configuring, 2-2
PostpaidService.properties file, 1-6, 1-7, 2-2
preferences, set, 19-15
preview
  audio MIME type, 15-2
  caption, 15-1
  device page, 19-19
  externally hosted, 15-8
price shown, 18-7
process
  log in, 19-12
  purchase, 19-17
  search for content, 19-14
  set preferences, 19-15
  view content, 19-13
  view My Downloads, 19-16
  view My Gifts, 19-17
  view My Wish List, 19-16
  view promotions, 19-16
ProcessOmaDrmMessageAdapter adapter, 14-2, 14-9
properties
  Catalog Manager, 18-12
  common, 18-2
  custom fields, 1-11
  Developer Portal, 18-3
  Device Client web services, 1-17
  DRM Fusion Toolkit, 4-2
  Monitoring Service, 1-2
  search result fields, 11-1
  Subscriber Portal, 18-4
  Vending Manager, 18-13
provisioning, automatic
  unknown user, 3-11
  unknown user peer, 1-18
publishing, automatic, 1-9
purchase
  confirm
    desktop, 18-6
    device, 18-7

process, 19-17
quick, 18-9
push adapter
CIMD2 properties, 8-5
definition, 8-1, 9-1
listener
overview, 9-1
register, 9-6
sender
overview, 8-1
register, 8-6
SMPP properties, 8-3
SMS HTTP properties, 8-4
push message
length, 18-8
mobile originated, 9-2
push proxy gateway, 13-5
push type, default, 18-5
push, enabling and disabling, 18-10, 18-11
PushListener.properties file, 9-4, 9-5
pushlistenerfactory.xml file, 9-6
pushsenderfactory.xml file, 8-6, 8-7

Q

quarantine a device, 18-9
quick browse, 18-9
quick purchase, 18-9

R

refresh period, external content, 18-3
registration
push listener adapter, 9-6
push sender adapter, 8-6
remote Vending Manager search fields, 11-2
reports, customized, 1-5
request headers, 1-18
response messages, saving, 10-2
restriction, gift, 18-8
result fields
Catalog Manager, 11-2
configuring, 11-1
Developer Portal, 11-5
properties, 11-1
Subscriber Portal, 11-7
Vending Manager, 11-5

S

sample
SMS push message template, 8-8
submission validation workflow criteria, 14-11
subsubmgr.xml file, 3-9
vsadminsubmgr.xml file, 3-10
schema.xml file, 11-7, 17-3
search
custom fields, 11-9
default fields, configuring, 11-8
default result fields
Catalog Manager, 11-2
Developer Portal, 11-5
Subscriber Portal, 11-7
Vending Manager, 11-5
field names
Catalog Manager, 11-3
Developer Portal, 11-3
Subscriber Portal, 11-5
Vending Manager, 11-5
remote Vending Manager fields, 11-2
search engine, configure, 11-7
security.config file, 3-2, 20-4
Separate Delivery, 13-5
server.policy file, 3-8
SetEditionWeightAdapter adapter, 14-2, 14-10
share, 19-19
share content expiration, 18-9
share content template, 18-9
signup link, 18-7
single sign-on, 6-1
SMPP, 8-3
push listener, 9-3
SMS push listener, 9-5
SMS push sender, 8-2
SMPP properties, 9-5
smppapi-0.3.7.jar file, 8-2
smppapi-0.3.7.tar.gz file, 8-2
SMS bundle, items in, 18-10
SMS HTTP, push sender, 8-4
SMS push
binary content, 8-6
messages, 8-2
SMS push sender adapter, 8-2
sms_push_msg_template.xsl file, 8-8
SMSC, 13-5

SMTP push sender adapter, 8-6
Solr, 11-7
`startApp` method, 13-4
`StockingWatermarking.properties` file, 15-4
streaming
  adapter, 5-2
  configuration, 5-1
  enable, 5-2
`Streaming.properties` file, 5-2
`struts-config.xml` file, 19-25
style sheet, 19-10
submission verifier workflow
  creating, 14-7
  default, 14-6
  externally hosted copyrighted content, 14-6
  iAppli applications, 14-5
  Java applications, 14-4
  specifying criteria, 14-11
  workflows provided, 14-3
subscriber account management, 3-8
subscriber adapter, LDAP, 3-1
subscriber integration, 3-1
subscriber notifications, opt-in, 20-4
Subscriber Portal
  device-specific page elements, 19-4
  generating pages for a device, 19-20
  modifying pages, 19-21
  pages, 19-2
  process
    log in, 19-12
    purchase, 19-17
    search for content, 19-14
    set preferences, 19-15
    view content, 19-13
    view My Downloads, 19-16
    view My Gifts, 19-17
    view My Wish List, 19-16
    view promotions, 19-16
  properties, 18-4
  style sheets for, 19-10
  user interface framework, 19-1
Subscriber Portal URL, 1-5
subscriber preferences, 20-4
subscriber provisioning, 3-11
`SubscriberPortal.properties` file, 1-17, 17-1, 18-4
`SubscriberPortalLocaleResource.properties` file, 15-1
synchronization, database, 1-4

T

tell a friend, 19-19
template, share content, 18-9
timeout, authorization key, 1-17
traps, 1-3
try content, 19-18
tuning LDAP, 3-8

U

`uname` command, 15-3
unprovisioned user, downloading to, 1-18
unsubscribe, 19-20
update license, 18-11
user interface framework, 19-1
User Profile API, 3-1

V

Vending Manage properties, 18-13
`VSAdminConsole.properties` file, 5-2, 12-2, 18-13

W

wait time, for LDAP, 3-3
WAP gateway adapter
  default, 7-2
  definition, 7-1
  Nokia Activ Server 2.0.1, 7-2
  Nokia Artus WAP gateway, 7-2
  Openwave WAP gateway, 7-2
WAP Gateway API, 7-1
WAP push sender adapter, 8-5
`wapgateway.config` file, 7-3
watermark
  default, 15-2
  define, 15-4
watermarking
  configuring, 15-2
  when previewing content, 15-8
  when stocking content, 15-4
WML-1_1.xsl, 19-11
WML-1_2.xsl, 19-11

workflow
See submission verifier workflow

X

XHTML-Alternate.xsl, 19-11
XHTML-AU.xsl, 19-11
XHTML-Basic.xsl, 19-11
XHTML-Color.xsl, 19-11
XHTML-IAppli.xsl, 19-11
XHTML-Motorola.xsl, 19-11
XHTML-NokiaSeries40.xsl, 19-11
XHTML-SE.xsl, 19-11
XHTML-Symbian.xsl, 19-11
XHTML-UP.xsl, 19-11
XML file
`pushlistenerfactory.xml`, 9-6
`pushsenderfactory.xml`, 8-7
`xml_record.xsl` file, 2-4