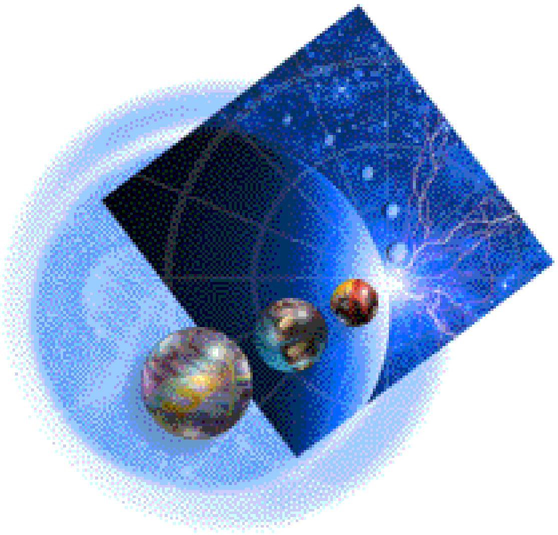
IBM Everyplace Intelligent Notification Services version 2.1



Everyplace Intelligent Notification Services

Everyplace Intelligent Notification Services

Everyplace Intelligent Notification Services delivers messages to users of pervasive devices based on the users' preferences and subscriptions. For example, subscribers can tell Everyplace Intelligent Notification Services to notify them when news articles with "pervasive computing" in the headline are published by a content provider. Subscribers can also specify which devices to send a notification to based on the urgency of the message. For example, if the message is marked FYI, send it via e-mail. If the message is marked urgent, send it via Sametime instant messaging.

Note: Click here for a set of steps to follow for setting up and running a notification system.

Customization

Intelligent Notification provides services for publishing content, subscribing to content, and sending simple notifications to other Everyplace Server users. Customization of the provided samples is required to tailor the services to your installation's needs. Content adapter samples illustrate how to publish content from various sources. Subscription samples show how to subscribe to and process content. Gateway adapter samples are also available for modification to support new gateway interfaces. All of the customizable interfaces to Intelligent Notification are described in the Application Programming Interface section. Refer to the Customize section for more details about which parts of Everyplace Intelligent Notification Services should be customized, what those parts do, and how to customize them.

Supported notification mechanisms

- Lotus Sametime
- Wireless Application Protocol (WAP)
- Short Message Service (SMS)
- SMTP e-mail

Related information

- Steps for setting up and running a notification system
- Customization overview

Everyplace Intelligent Notification Services Components

Everyplace Intelligent Notification Services consists of several components that enable the delivery of messages to subscribed users.

Click here for an overview diagram of Everyplace Intelligent Notification Services.

Universal Notification Dispatcher

The Universal Notification Dispatcher (UND) delivers messages to users based on their preferences. Messages may originate as simple notifications from other users or applications, or they may result from content subscriptions. Messages can be delivered via several mechanisms: WAP, SMS, e-mail, and Sametime.

iQueue Server

The iQueue Server accepts content from external sources, such as news feeds from Reuters. Content is published to the iQueue Server via content adapter applications that process the data feeds. The iQueue Server matches content to user subscriptions and sends notifications to subscribers. Content may also be optionally stored in a database for subsequent retrieval by a subscriber.

Intelligent Notification Application Programming Interface

The Intelligent Notification Application Programming Interface (API) provides access to services used by notification applications. The API is used by the following types of applications.

- *Subscriptions* allow users to specify content sources and filters.
- *Trigger handlers* process subscription specifications.
- *Content adapters* process data feeds and publish the content to the iQueue Server for subscription matching.
- *Simple notification applications* send messages from other Everyplace Server users or applications.

Sample source code

Sample code is available for each of these applications. Source code is also provided for the gateway adapters. The gateway adapters define the interfaces to the Everyplace Wireless Gateway for WAP, SMS, and SMTP e-mail notifications. Additionally, there are gateway adapters for interfacing with Sametime and SMTP servers.

The following links provide more detailed information about customizing the gateway adapters.

- Gateway adapters
- Transcoders
- Transcoder stylesheets

Related Components

Everyplace Setup Manager installs and configures Everyplace Intelligent Notification Services. When you select Intelligent Notification to be installed, Setup Manager installs the following required programs.

- WebSphere Application Server, Advanced Edition
- SecureWay Directory
- DB2 UDB, Enterprise Edition

Everyplace Wireless Gateway is optionally installed if you plan to use the WAP, SMS, and e-mails gateways supplied with this program. It is recommended that Everyplace Wireless Gateway be used for these notification mechanisms.

Everyplace Suite Manager is installed by Setup Manager and may be used to start and stop the UND and iQueue servers.

WebSEAL-Lite is also installed by Setup Manager and provides user authentication services.

Related information

- Graphical overview of the Everyplace Intelligent Notification Services system
- iQueue Server
- Universal Notification Dispatcher
- Gateway Adapters
- End user servlets

Everyplace Intelligent Notification Services - Components - iQueue Server

The iQueue Server interacts with the Universal Notification Dispatcher, content adapters, and subscription information. Content adapters publish content into the iQueue system from an external data feed or source. The subscriptions specify content topics and filters from the user to the iQueue Server, so it knows what information items the user is interested in. When the iQueue Server finds information of interest to the user, it sends out a notification to the user via the Universal Notification Dispatcher.

Triggers define the specifications for a user's subscription. Subscription specifications include items such as content source, content filter, and a unique identifier. Triggers also define a "handler" program that processes content when a match occurs. The iQueue Server has a Trigger Manager that keeps track of trigger configurations and activations. When a trigger is activated, it executes a TriggerHandler. Several TriggerHandlers are defined in the Everyplace Intelligent Notification Services Subscription sample. Click here for more details on how to customize a Trigger Handler for your installation.

Sometimes an iQueue application will store content that is too lengthy to include in a notification message. The end user may later retrieve the stored content. For instance, a news article of interest to the subscriber could contain too much text to send as a Sametime message or to a WAP-enabled phone. So instead, the TriggerHandler requests that the iQueue Server store the news article in its database. The iQueue Server then sends a message via the Universal Notification Dispatcher informing the subscriber that the article exists and provides the user with a URL and an article ID number with which to retrieve the article from the database.

Related information

- How to customize a subscription application, including the Trigger Handler class.

Everyplace Intelligent Notification Services - Components - Universal Notification Dispatcher

The Universal Notification Dispatcher receives notification requests from the iQueue Server. The Universal Notification Dispatcher can also receive requests from an application, bypassing the iQueue Server. The Universal Notification Dispatcher sends that message out to the appropriate gateway adapter. The Wireless Gateway then sends the notification messages out to a user's devices based upon the preferences the user has set with User Preferences.

User Preferences allows the user to specify user, group, and device information. Universal Notification Dispatcher uses these settings to determine where to send the notification message. The preferences must be set before any messages can be sent. For instance, at least one user must be defined. At least one device must be defined for that user. At least one device must be specified to receive messages from the anonymous group. Click here for more information about User Preferences.

Related information

- User Preferences

Everyplace Intelligent Notification Services - Supported devices, gateway adapters, and transcoders

Supported notification mechanisms

The following notification mechanisms are supported by the Everyplace Wireless Gateway and the Everyplace Intelligent Notification Services.

- Wireless Application Protocol (WAP)
- Lotus Sametime
- Short Messaging Service (SMS)
- SMTP e-mail

Gateway adapters

Gateway Adapters connect the Universal Notification Dispatcher to the Everyplace Wireless Gateway and any other gateways being used. Each device type requires a gateway adapter.

Click here to view a graphical representation of gateway adapters.

Transcoders

Each gateway adapter has a transcoder that formats notification messages for the target device. Transcoders use stylesheets for formatting if one is available. Modify the sample Transcoder stylesheet for a particular device to customize the formatting of messages for that device.

Gateway adapter and transcoder configuration in SecureWay Directory

Gateway adapters and transcoders are registered in SecureWay Directory. Based on the name and location settings stored in SecureWay Directory, the Universal Notification Dispatcher selects the appropriate adapter for the recipient device. SecureWay Directory stores the gateway adapter and transcoder names and locations. View or modify the gateway adapter and transcoder properties in SecureWay Directory using the Directory Management Tool.

- Click here for more information on the Directory Management Tool.
- Click here for more information on how to edit gateway adapter and transcoder properties.

Customizable code

The transcoders provided should cover most scenarios. However if different or additional function is required, the transcoders may be modified, or the GenericTranscoder extended.

We highly recommend using Everyplace Wireless Gateway. However, the Everyplace Intelligent Notification Services gateway adapters may be modified to work with other gateways. Refer to the Gateway adapter part of the Customize section for more information on how to customize the default gateway adapters.

Refer to the Customize section to view the gateway adapter, transcoder, and transcoder stylesheet code.

Related information

- Overview diagram of gateway adapters
- Administering properties with the SecureWay Directory Management Tool
- Customizing Gateway Adapters
- Customizing Transcoders
- Customizing Transcoder Style Sheets

Everyplace Intelligent Notification Services - subscriber servlets

There are three types of JSPs and servlets the subscriber interacts with in Everyplace Intelligent Notification Services.

- User Preferences
- Subscriptions
- Retrieve persistent messages

User Preferences

The User Preferences run with Tivoli Personalized Subscription Manager (TPSM). The servlets store preference information in SecureWay Directory. The information collected and maintained here is later used by the Universal Notification Dispatcher (UND) to figure out where to send messages.

With User Preferences, a user can define devices, device properties, groups, and group access to their defined devices. For information on how to access User Preferences, refer to the User Preferences documentation in the Everyplace Server documentation.

The user preference JSPs and servlets allow the end user to perform self care of their subscription and the administrator to perform customer care for the customers subscribed to their system. Refer to the User Preferences and TPSM documentation for more information on self care and customer care.

Subscriptions

Subscription JSPs and servlets are distributed as samples that can be modified to implement a customized subscription solution. The subscription JSPs and servlets run on WebSphere Application Server. These servlets collect information from the user about their subscription preferences, for example, stock names, high and low stock prices, key words to look for in headlines, and whether or not to save a matching content for later retrieval.

Retrieve persistent messages

ContTransServlet.java and ContTransServlet.class

The retrieve persistent messages servlet allows the user to go to a URL and view persistently stored content on a Web browser.

For example, a user subscribes to a news service and asks to be notified when a story containing the string "pervasive computing" is published. When the iQueue Server detects a matching story, it sends the story content to a database for storage and sends the user a notification message via the Universal Notification Dispatcher. The notification message contains a URL for the story and perhaps an abbreviated headline. The user specifies the URL in a Web browser. The full story text appears in the Web browser.

Related information

- Everyplace Intelligent Notification Services User Preferences

Everyplace Intelligent Notification Services - subscriber scenarios

The following are three scenarios that illustrate how one might subscribe to notification services. Each scenario has a corresponding subscription sample. Refer to the samples section for more information about the subscription samples.

- Stock scenario
- News scenario
- Weather scenario
- News, stock, and weather subscription samples

Stock scenario

The subscriber wants to be able to check stock information at crucial points during the market day, but she is often travelling for her job. She subscribes to stock content, provided by an external source, which is matched and filtered by Intelligent Notification according to her subscription. Stock information is delivered to her notification devices according to her preferences.

The subscriber registers for the service at a Web site with a JSP hosted by WebSphere Application Server. She specifies which stocks to watch, high and low stock prices for each, and preferences for receiving notification.

News scenario

The subscriber wants to find articles about "pervasive computing" without surfing the Web or checking his e-mail all the time. He subscribes to news content, provided by an external source, which is matched and filtered by Intelligent Notification according to her subscription. News information is delivered to her notification devices according to her preferences.

The subscriber registers for the service at a Web site with a subscription JSP hosted by WebSphere Application Server. He specifies matching criteria for articles such as news topics of interest and article title substrings.

Weather scenario

The subscriber wants to be aware of weather events. She wants to receive weather information for her geographic location only. She does not want to go to a weather Web site and search for her location-specific weather information.

The subscriber subscribes to weather content, provided by an external source, which is matched and filtered by Intelligent Notification according to her subscription. She specifies her city and state and whether she would like to receive information about current conditions, future conditions, or a full weather report. Weather information is delivered to her notification devices according to her preferences.

News, Stock, and Weather subscription samples

The Subscription samples demonstrate how to implement these scenarios with Everyplace Intelligent Notification Services. Click here for instructions on how to get these samples running.

Related information

- Subscription samples

Everyplace Intelligent Notification Services - planning steps

Note: The planning steps assume that you have installed Intelligent Notification with Setup Manager and started the servers as described in the Start and stop servers section.

Click here for directions on how to start the Intelligent Notification Services.

1. **Select your content sources and set up an interface to them.** Content provider services and Web pages are examples of content source you can use with Everyplace Intelligent Notification Services.

   Subscribe to a content provider service.

   Subscribe to a content provider service like iSyndicate, Reuters, or MarketWatch.com. Download and install the content provider's client application. Customize the ContentAdapter sample or write a new ContentAdapter to obtain content from the content provider's client application and publish the content to the iQueue Server.

   Write a Web page parser.

   Write an application to parse content from a Web page. For example, an application might periodically parse stock information from a Web site that posts stock quotes with timed updates. The application may parse current weather information every hour from a weather Web site. Customize the ContentAdapter sample or write a new ContentAdapter to obtain the parsed content from the Web page parser and publish the content to the iQueue Server.

   Click here for more information about how to customize or write a ContentAdapter.

2. **Set up user authentication with WebSEAL-Lite or WebSphere Application Server.**

   - Set the `iqAuthFlag` property to `TRUE` in the *IQ.properties* file.
   - Configure WebSEAL-Lite or WebSphere Application Server to perform authentication for Everyplace Intelligent Notification Services. Refer to the WebSEAL-Lite or WebSphere Application Server documentation for instructions on how to configure for authentication.
   - Customize the TriggerHandler code to specify to the user the correct URL, one that goes through the authentication server. Click here for more information on how to customize the TriggerHandler.

3. **Define users, their devices, and notification groups.** Use User Preferences to set up a set of Everyplace Intelligent Notification Services users. Define users, devices, and notification groups. Use either the graphical user interface to edit single entries, or use the script file to perform batch modifications.

Use the graphical user interface.

- ❍ Add users.
- ❍ Set user preferences.
- ❍ Add user devices.
- ❍ Set device preferences.
- ❍ Create groups for users.
- ❍ Set notification preferences and member lists for groups.

Edit User Preferences with an LDIF file

An LDIF file can be used to make batch modifications to User Preferences. Refer to User Preferences for information on how to create an LDIF file for this purpose.

4. **Subscribe to content sources.** Set up user subscriptions using the sample subscription applications or customized versions of the sample applications.

   Click here for more information on how to use the sample subscription applications to set up user subscriptions.

   Click here for more information on how to customize the sample subscription applications.

5. **Run the ContentAdapter application.** Run either XMLContentAdapter or a customized version of the ContentAdapter.

   Click here for instructions on how to run the ContentAdapter application. Refer to step #3.

   Click here for instructions on how to customize the XMLContentAdapter sample.

Reasons for message delivery failure: Notification messages may fail to be sent for a variety of reasons. The following list gives examples of possible reasons for delivery failure.

- The recipient is offline for instant messaging.
- No devices were specified for the recipient.
- The message sender is specified as part of a group that does not have access to the recipient's devices.

Refer to the NotificationService class documentation in the API Javadoc for a full explanation of sendMessage() function return values. The return values specify reasons for delivery failure.

If the trigger notification preference is set to "Always," delivery attempts will continue to be made, despite previous failed delivery. If the trigger notification preference is set to "Once," no additional delivery attempts will be made.

Related information

- How to customize a ContentAdapter
- How to add users and edit user preferences
- How to start and stop the Everyplace Intelligent Notification Services
- How to set up user subscriptions with the subscription sample
- How to customize the sample subscription applications
- How to run the XMLContentAdapter

Everyplace Intelligent Notification Services - prerequisites and limitations

Required applications:

The following programs are installed by Setup Manager when Intelligent Notification is installed.

- WebSphere Application Server, Advanced Edition
- SecureWay Directory
- DB2 UDB, Enterprise Edition

Everyplace Wireless Gateway for WAP, SMS, and e-mail gateways is optionally installed. It is recommended that Everyplace Wireless Gateway be used for these notification mechanisms.

Note: Follow the installation and configuration instructions for SecureWay Directory and Tivoli Personalized Services Manager. These directions are located in the WebSphere Everyplace Server documentation in the Getting Started > Steps section. Follow steps one and two. Step one is how to set up SecureWay Directory. Step two is how to set up Tivoli Personalized Services Manager.

Other requirements

- A content provider will also be needed to implement a subscription solution. The content provider provides a client application that is connected to an Internet data feed. The content adapter interfaces with this application to retrieve data to publish to the iQueue Server.
- The samples for this product require JSP 1.1 support. Ensure that a browser with support for JSP 1.1 is available for viewing the sample front-ends.
- Everyplace Intelligent Notification Services requires at least 7MB of permanent disk space. Additional disk space may be needed to store XML content files from the data feeds.

Sametime Server limitations

- Everyplace Intelligent Notification Services can only point to one Sametime Server.

ContentBody limitations

- ContentBody is the class that holds a string representation of XML-formatted content published to the iQueue Server by a ContentAdapter. This string representation of the content must be less than or equal to 64 KB in length. To publish content greater then 64 KB in length requires breaking the content into 64 KB segments.

SCS-related error messages

Ignore any error messages relating to "SCS" or "calendar context." These errors relate to future functionality that has not been implemented for this version. For example, ignore the following error messages.

```
IBM Intelligent Notification Service UND Could not establish link to SCS

configuration file contains poorly formated port number: {0}

no calendar context found
```

Everyplace Intelligent Notification Services - Use Setup Manager.

Everyplace Intelligent Notification Services is installed by Everyplace Setup Manager when selected from the Everyplace Setup Manager. Everyplace Setup Manager installs all of the prerequisite software for Everyplace Intelligent Notification Services. During the installation, Everyplace Setup Manager prompts for configuration information. Everyplace Intelligent Notification Services requires at least 7MB of disk space.

Refer to the Install section of the WebSphere Everyplace Server documentation for more details on how to use the Everyplace Setup Manager.

Sametime Server limitations

Only one Sametime Server can be defined to Everyplace Intelligent Notification Services.

Everyplace Intelligent Notification Services - configure

- Configure for authentication.
- Configure for e-mail notification.
- Configure for SMTP e-mail notification.
- Configure for Wireless Gateway Push e-mail notification.

Configure for authentication using WebSphere Application Server or WebSEAL-Lite.

- Set the `iqAuthFlag` property to `TRUE` in the *IQ.properties* file.
- Configure WebSEAL-Lite or WebSphere Application Server to perform authentication for Everyplace Intelligent Notification Services. Refer to the WebSEAL-Lite or WebSphere Application Server documentation for instructions on how to configure for authentication.
- Customize the TriggerHandler code to specify to the user the correct URL, one that goes through the authentication server. Click here for more information on how to customize the TriggerHandler.

Configure Everyplace Intelligent Notification Services for e-mail notification.

Everyplace Intelligent Notification Services can be configured for one of two types of e-mail notification. Each type of e-mail notification requires different configuration steps to be taken.

- Configure for SMTP e-mail notification.
- Configure for Wireless Gateway Push e-mail notification.

Configure for SMTP e-mail notification.

1. Ensure that the `ibm-undDeviceAdaptorLink` is set to `com.ibm.pvc.we.ins.und.server.adaptors.Mail`. This setting tells the Universal Notification Dispatcher to use the *Mail.class* gateway adaptor from the `com.ibm.pvc.we.ins.und.server.adaptors` package.

   The entry, `ibm-undDeviceAdaptorLink`, is located in the directory tree under `sys=und`.

2. Configure Everyplace Wireless Gateway to support an SMTP server. Refer to the Everyplace Wireless Gateway information for instructions on how to configure the Gateway to support an

SMTP server.

1. Define a Wireless Gateway using the Wireless Gatekeeper.
2. Add a messaging gateway to the Wireless Gateway.
3. Configure the messaging gateway to connect to the SMTP e-mail server.

The messaging gateway will then be able to forward short messages to the end user that can be addressed with an e-mail address.

Configure for Wireless Gateway Push e-mail notification.

1. Ensure that the `ibm-undDeviceAdaptorLink` is set to `com.ibm.pvc.we.ins.und.server.adaptors.MailPush`. This setting tells the Universal Notification Dispatcher to use the *MailPush.class* gateway adaptor from the `com.ibm.pvc.we.ins.und.server.adaptors` package.

   The entry, `ibm-undDeviceAdaptorLink`, is located in the directory tree under `sys=und`.

2. No additional wireless gateway configuration is required.

Everyplace Intelligent Notification Services - start and stop servers

- Start the JDBC driver.
- Start the servers with the Suite Manager.
- Start the servers manually.

Start the JDBC driver

Be sure to start the JDBC driver before starting the Intelligent Notification servers. Start the JDBC driver on the computer on which DB2 is installed.

1. Manually start the JDBC driver.
   1. Switch to the Intelligent Notification user instance. Type in `su - <user instance>`, where `<user instance>` is the user instance for Everyplace Intelligent Notification Services. `insdb2` is the default value for the Intelligent Notification database user instance.
   2. Start the driver on port 6789. This port is the designated, hard-coded port for the JDBC driver. You must use port 6789. Type in `db2jstrt 6789`.
2. Verify that the driver has started correctly.
   1. Search for the process by typing `ps -ef |grep -i db2jd`. If the process `db2jd` is found, it is running.
   2. Check to see that the port 6789 is active. Type `netstat -a |grep 6789`. When the port status information appears, check for the word "listen."

Start the servers with the Suite Manager.

Right click on the service icon to pop up a task menu to start and stop Universal Notification Dispatcher or iQueue Server. In the "View by component" view, the Everyplace Intelligent Notification Services subtree appears in the following manner.

```
Everyplace Intelligent Notification Services
        |--- --- IQueue Server
        |--- --- Universal Notification Dispatcher
```

Note: Universal Notification Dispatcher MUST be started before iQueue Server. Upon starting, iQueue Server tries to connect to the Universal Notification Dispatcher and the content matching engine. If either the Universal Notification Dispatcher or the content matching engine is not available, the iQueue Server issues an error message. The content matching engine is started automatically when the iQueue Server is started from Suite Manager.

Refer to the Suite Manager section of the general Everyplace Server information for more details on how to use the Suite Manager.

Start the servers from a command prompt.

Start servers from the Everyplace Intelligent Notification Services bin directory.

- `/usr/IBMEPS/INS/bin` on AIX
- `/opt/IBMEPS/INS/bin` on Solaris

Enter the following commands to start each server. The servers must be started in the following order.

1. Type `./startUND <host name>` to start the Universal Notification Dispatcher
2. Type `./startGB` to start the content matching engine
3. Type `./startIQ <host name>` to start the iQueue Server

`<hostname>` is the name of a Server System entry in LDAP identified by a dc= name. For example, if the user configured iQueue Server to run on a host named iowa, the Setup Manager would create a Server System entry in LDAP identified by dc=iowa. When iQueue Server was started, the command would be: `./startIQ iowa`.

Note: Universal Notification Dispatcher and the content matching engine MUST be started before iQueue Server. Upon starting, iQueue Server tries to connect to the Universal Notification Dispatcher and the content matching engine. If either the Universal Notification Dispatcher or the content matching engine is not available, the iQueue Server issues an error message.

Everyplace Intelligent Notification Services - message logging

Turn logging on and off.

Turn console logging and file logging on or off for the following services.

- iQueue Server
- Universal Notification Dispatcher
- Preferences

Use the SecureWay Directory Management Tool (DMT) to modify the appropriate system properties. For a list of logging system properties, refer to the properties chart in the section below.

Note: Restart the corresponding server after modifying logging settings. The Everyplace Intelligent Notification Services system reads in logging settings only at start time.

View message logs.

Everyplace Intelligent Notification Services generates iQueue Server logs, Universal Notification Dispatcher logs, and Directory Services logs.

File logging
Turn file logging on by setting the file logging properties in SecureWay Directory by ensuring that there is a file name specified for the file logging properties. Use the chart below to see which SecureWay Directory property entries store the log file names. Take note of the names of the log files. Restart the corresponding server, and perform the actions to be logged. Then use any text editor to view the log files.

Log files without a directory specified are stored in the `IBMEPS/INS/bin` directory. Specify a fully qualified file path to store the log files elsewhere.

Console logging
Turn console logging on by setting the console logging properties in SecureWay Directory to "true." Use the chart below to see which SecureWay Directory property entries to set. Restart the corresponding server, and perform the actions to be logged. View the log messages as they are displayed in the command console.

Locate and modify logging properties.

For each domain, the Everyplace Intelligent Notification Services properties are located in the directory tree under `sys=SDP`. Then, under `sys=SDP`, the properties are grouped by service.

- The iQueue Server general properties are under `sys=ins`.
- The Universal Notification Dispatcher properties are under `sys=und`.

- The Preferences are under `sys=pref`.

Logging properties chart:

Note: In addition to the logging properties, there are several other SecureWay Directory entries that may be modified. For a table of these additional modifiable SecureWay Directory entries, refer to the Administer > Properties in Directory Management Tool section. Do not modify the other SecureWay Directory entries. Keep the default values for all other SecureWay Directory entries.

Service name and abbreviation	SecureWay Directory property entry name	description	Default value
iQueue Server general properties abbreviation: *ins*	`ibm-insSystemLogger.console`	"true" or "false" whether to send messages to the command console.	false
	`ibm-insSystemLogger.file`	name of the file to store messages. If this value is null, messages will not be logged to a file.	null
	`ibm-insTraceLogger.console`	"true" or "false" whether to send trace messages to the command console.	false
	`ibm-insTraceLogger.file`	name of the file to store trace messages. If this value is null, trace messages will not be logged to a file.	null
Universal Notification Dispatcher abbreviation: *und*	`ibm-undConsoleLogOutput`	"true" or "false" whether to send und messages to the command console.	false
	`ibm-undLogFileLocation`	name of the file to store und messages. If this value is null, und messages will not be logged to a file.	null
	`ibm-undConsoleTraceOutput`	"true" or "false" whether to send und trace messages to the command console.	false
	`ibm-undTraceFileLocation`	name of the file to store und trace messages. If this value is null, und trace messages will not be logged to a file.	null
	`ibm-prefMessagesLogToConsole`	"true" or "false" whether to send pref messages to the command console.	false

Preferences abbreviation: *pref*	`ibm-prefMessagesLogFile`	name of the file to store pref messages. If this value is null, pref messages will not be logged to a file.	null
	`ibm-prefTracesLogToConsole`	"true" or "false" whether to send pref trace messages to the command console.	false
	`ibm-prefTracesLogFile`	name of the file to store pref trace messages. If this value is null, pref trace messages will not be logged to a file.	null

SCS-related error messages

Ignore any error messages relating to "SCS" or "calendar context." These errors relate to future functionality that has not been implemented for this version. For example, ignore the following error messages.

```
IBM Intelligent Notification Service UND Could not establish link to SCS

configuration file contains poorly formated port number: {0}

no calendar context found
```

Everyplace Intelligent Notification Services - User Preferences

Overview

With User Preferences, a user can define devices, device properties, groups, and group access to the defined devices. For information on how to access User Preferences, refer to the User Preferences documentation in the Everyplace Server documentation.

Note: Everyplace Intelligent Notification Services User Preferences is part of WebSphere Everyplace Server User Preferences, which runs on Tivoli Personalized Services Manager. Make sure that Tivoli Personalized Services Manager is properly installed, configured, and running. For instructions on how to install and configure Tivoli Personalized Services Manager, refer to the Getting started > Steps section of the WebSphere Everyplace Server documentation.

Devices

A user must define a device and specify device properties for each device with which he or she would like to receive notification messages, for example e-mail, WAP, Sametime, or SMS. Each e-mail address would require a separate device entry. Each WAP device requires a separate device entry.

Groups

Groups categorize users and give them specified types of access to certain devices. A user adds other Everyplace users to one or more of his or her groups. By default, a group-less user attempting to notify is classified as belonging to the anonymous group. So if a user has not added another user to one of their groups, the other user will have only the notification access given to the anonymous group.

Edit User Preferences with GUI

A user can also be created through Tivoli Personalized Services Manager and edited via the user self-care preference jsps or via the SecureWay Directory's Directory Management Tool (DMT). A browser is needed to modify user information with the GUI. The GUI is useful in making the following type of modifications to User Preferences.

Add device

Click on the "Edit Base Preferences" item in the left-hand navigation frame. Add a device from the Edit Base Preferences page.

Edit device

Click on the "Edit INS Devices" item in the left-hand navigation frame. This will bring up a list of current devices. Select a device to edit. This will bring up the editable properties page for the selected device. Edit the desired property settings.

Delete device

Click on the "Edit INS Devices" item in the left-hand navigation frame. This will bring up a list of current devices. Delete the desired device from this page.

Add group

Click on the "Edit INS Groups" item in the left-hand navigation frame. This will bring up a list of current groups. Type in the name of the new group and click the Add button. This will bring up a panel to specify device access levels and a user list for the group. Specify which devices the group will have notification access to, and for each device, specify whether to receive FYI, normal, or urgent messages from the users in the group. Then add users to the group. In order to add a user to the group, the user must be a registered Everyplace user. Use the Everyplace userid to add the user to the group. User enrollment in Everyplace is done in Tivoli Personalized System Manager. Refer to the Tivoli Personalized System Manager documentation for more information on user enrollment.

Edit group

Click on the "Edit INS Groups" item in the left-hand navigation frame. This will bring up a list of current groups. Select a group to edit. This will bring up the editable properties page for the selected group. Edit the notification access and user list for the group.

Delete group

Click on the "Edit INS Groups" item in the left-hand navigation frame. This will bring up a list of current groups. Delete the desired group from the list.

Edit User Preferences with LDIF

An alternate method that is more useful for batch edits is to create and import an LDIF file. An LDIF file is a text representation of one or more LDAP entries and can be imported into the LDAP directory with the `ldapadd` or `ldapmodify` command line executable, both installed with the LDAP server. See the LDAP server documentation for the syntax of the commands.

Create an LDIF file

Below is a sample LDIF file containing the entries for one user. An LDIF file can contain any number of users. An INS user entry consists of the following components:

- exactly one `ibm-SdpUser` entry
- exactly one `ibm-undUser` entry
- exactly one `groupOfNames` with `cn`=anonymous, along with a corresponding `ibm-undGroup` sub-entry
- may contain one of the `groupOfNames` objects, along with a corresponding `ibm-undGroup`

sub-entry

- may contain ibm-device entries and corresponding ibm-undDevice sub-entry

User Entry

The dn= is composed of the userid (uid), the user's realm, and the LDAP servers root suffix. The user id and other attributes should be consistent throughout the entry. Use one line for each device. The value of each ibm-deviceidlist line must be the deviceid attribute of the ibm-device entry below. See below for an example of a user or ibm-sdpuserentry where joeuser92@myrealm.ibm.com is the user.

```
dn: uid=joeuser92,cn=users,ou=myrealm.ibm.com, \
\ dc=raleigh,dc=ibm,dc=com
objectclass: ibm-CertificateForDN
objectclass: inetOrgPerson
objectclass: ibm-SdpUser
objectclass: ibm-deviceList
objectclass: ePerson
objectclass: cimManagedElement
objectclass: eUser
objectclass: organizationalPerson
objectclass: person
objectclass: top
uid: joeuser92
businesscategory: MAN101
employeetype: 13
ibm-tismstatus: C
homephone: 6665555
mobile: 4443332222
postalcode: 27707
ou: myrealm.ibm.com
cn: Joe User
initials: W
telephonenumber: 5556666
employeenumber: 10514141859896605
st: NC
l: RTP
postaladdress: 123 High House
sn: User
ibm-deviceidlist: mail1
ibm-deviceidlist: phone1
ibm-deviceidlist: sametime1
ibm-certificatesubjectandissuer::
title: None
```

The next entry is the ibm-unduser entry. The dn should be identical to the ibm-sdpuser entry in the above example with cn=username/und appended to the front. Add one line for each device where the value should be in the form of cn=deviceid/und. Add one line for each group where the value should be cn=groupname/und.

```
dn: cn=joeuser92/und,uid=joeuser92,cn=users, \
\ ou=myrealm.ibm.com,dc=raleigh,dc=ibm,dc=com
ibm-deviceidlist: cn=mail1/und
```

```
ibm-deviceidlist: cn=phone1/und
ibm-deviceidlist: cn=sametime1/und
objectclass: ibm-undUser
objectclass: cimManagedElement
objectclass: eUser
objectclass: top
ibm-grouplist: cn=anonymous/und
ibm-grouplist: cn=friends/und
cn: joeuser92/und
```

Group entry

In the group entry, the dn is the same as that of the `ibm-spduser` where `cn=groupname` is appended to the front. Add one line for each member of the group where the value should be equal to the uid of the person's LDAP entry. cn is the base name of the group.

```
dn: cn=anonymous,uid=joeuser92,cn=users, \
\ ou=myrealm.ibm.com,dc=raleigh,dc=ibm,dc=com
objectclass: groupOfNames
objectclass: top
member:uid=jane@myrealm.ibm.com
member:uid=john54@myrealm.ibm.com
cn: anonymous
```

This is the `ibm-undgroup` entry. The dn should be identical to that of the `groupOfNames` where `cn=groupname/und` is appended to the front. Add one or more lines for each message priority for each device you want to use to receive messages. In the example below, the mail1device will receive messages of priority normal, and the phone1 device will receive only messages of urgent priority, and no devices will receive FYI messages.

```
dn: cn=anonymous/und,cn=anonymous,uid=joeuser92, \
\ cn=users,ou=myrealm.ibm.com,dc=raleigh,dc=ibm,dc=com
objectclass: ibm-undGroup
objectclass: top
ibm-undnormaldevicesauthorizations:mail1
ibm-undurgentdevicesauthorizations:phone1
ibm-undurgentdevicesauthorizations:mail1
ibm-undfyidevicesauthorizations::
ibm-undauthorizations::
cn: anonymous/und

dn: cn=friends,uid=joeuser92,cn=users,ou=myrealm.ibm.com, \
\ dc=raleigh,dc=ibm,dc=com
objectclass: groupOfNames
objectclass: top
cn: friends
member: uid=michael@yahoo.com
member: uid=john533@hotmail.com

dn: cn=friends/und,cn=friends,uid=joeuser92,cn=users, \
\ ou=myrealm.ibm.com,dc=raleigh,dc=ibm,dc=com
ibm-undnormaldevicesauthorizations: mail1
```

```
ibm-undnormaldevicesauthorizations: sametime1
ibm-undnormaldevicesauthorizations: phone1
ibm-undauthorizations::
objectclass: ibm-undGroup
objectclass: top
ibm-undurgentdevicesauthorizations: mail1
ibm-undurgentdevicesauthorizations: phone1
ibm-undurgentdevicesauthorizations: sametime1
ibm-undfyidevicesauthorizations: sametime1
ibm-undfyidevicesauthorizations: mail1
cn: friends/und
```

Devices

This is the `ibm-device` entry. The dn should be identical to that of the `ibm-sdpuser`. The deviceID is the identifier of the device. If the device has an MIN, MSISDN, or IP Address, then this should be the deviceID because it is used by the some Everyplace services to identify a device. The following is an example of a device entry. Indented lines that begin with a pound # sign provide additional comments about the previous line.

```
dn: deviceID=mail1,uid=joeuser92,cn=users,ou=myrealm.ibm.com,
dc=raleigh,dc=ibm,dc=com
deviceID: mail1
objectclass: ibm-device
objectclass: cimLogicalElement
objectclass: cimManagedElement
objectclass: cimManagedSystemElement
objectclass: cimLogicalDevice
objectclass: top
ibm-isdeviceenabled: true
   # Indicates if the device is useable. True or False.
description: work email
   # Description provides user-friendly identifier for the device
dn: cn=mail1/und,deviceID=mail1,uid=joeuser92,cn=users,
ou=myrealm.ibm.com,dc=raleigh,dc=ibm,dc=com
   # The ibm-unddevice entry. The dn should be identical to that of the ibm-device where cn=deviceid/und
is appended to the front
ibm-appprotocolversion: none
   # Currently used only by IM devices and is the protocol version. Otherwise, use 'none'.
ibm-appdeviceaddress: joeuser92@myrealm.ibm.com
   # the address used to deliver messages to the device. In this case, an email address.
objectclass: ibm-undDevice
objectclass: top
uid: none
   # uid is not currently used, but future gateway adapters may need to authenticate
   # a device gateway with an id and password before a message can be sent.
cn: mail1/und
   # The cn of the base device with /und appended
description: work email
ipserviceport: 25
   # The IP port that the host server listens on to receive message request.
host: mymailserver.ibm.com
   # The hostname of the server which delivers messages to this device.
ibm-appprotocoltype: mail
   # The type of message. Valid values are IM, mail, WAP, and SMS.
```

```
dn: deviceID=sametime1,uid=joeuser92,cn=users,ou=myrealm.ibm.com, \
\ dc=raleigh,dc=ibm,dc=com
deviceid: sametime1
objectclass: ibm-device
objectclass: cimLogicalElement
objectclass: cimManagedElement
objectclass: cimManagedSystemElement
objectclass: cimLogicalDevice
objectclass: top
ibm-isdeviceenabled: true
ibm-deviceidtype: 2
description: work sametime

dn: cn=sametime1/und,deviceID=sametime1,uid=joeuser92, \
\ cn=users,ou=myrealm.ibm.com,dc=raleigh,dc=ibm,dc=com
ibm-appprotocolversion: V15
ibm-appprotocol: sametime
ibm-appdeviceaddress: joeuser92OnSametime
objectclass: ibm-undDevice
objectclass: top
cn: sametime1/und
description: work sametime
ipserviceport: 1553
host: sametime.ibm.com
ibm-appprotocoltype: IM

dn: deviceID=5555551000,uid=joeuser92,cn=users,ou=myrealm.ibm.com, \
\ dc=raleigh,dc=ibm,dc=com
deviceid: 5555551000
objectclass: ibm-device
objectclass: cimLogicalElement
objectclass: cimManagedElement
objectclass: cimManagedSystemElement
objectclass: cimLogicalDevice
objectclass: top
ibm-isdeviceenabled: false
ibm-deviceidtype: 2
description: wap phone

dn: cn=5555551000/und,deviceID=5555551000,uid=joeuser92,cn=users, \
\ ou=myrealm.ibm.com,dc=raleigh,dc=i
bm,dc=com
ibm-appprotocolversion: none
ibm-appprotocol: none
ibm-appdeviceaddress: 5555551000
objectclass: ibm-undDevice
objectclass: top
cn: 5555551000/und
description: wap phone
```

```
ipserviceport: 0
host: <yourhost>
ibm-appprotocoltype: wap
```

Everyplace Intelligent Notification Services - properties

- Maintaining Everyplace Intelligent Notification Services properties with the SecureWay Directory Management Tool
- Locating properties in the Directory Management Tree
- Properties charts

Maintaining Everyplace Intelligent Notification Services properties with the SecureWay Directory Management Tool

Everyplace Intelligent Notification Services properties are not accessible through the Everyplace Suite Manager console. To track and modify these property settings, use the SecureWay Directory Management Tool. The SecureWay Directory Management Tool displays all information that is made available to it by SecureWay Directory. The SecureWay Directory Management Tool is installed automatically with SecureWay Directory.

Locating properties in the Directory Management Tree

For each domain, the Everyplace Intelligent Notification Services properties are located in the directory tree under `sys=SDP`. Then, under `sys=SDP`, the properties are grouped by service.

- The Notification Services general properties are under `sys=ins`.
- The Universal Notification Dispatcher properties are under `sys=und`.
- The Preferences properties are under `sys=pref`.

- System properties - general

setting ID	description	values
ibm-insJDBCDatabaseURL	JDBC url string	Host name where DB2 is installed, port # for DB2 JDBC daemon, and iQueue Server database name. JDBC daemon default port = 6789 Example: jdbc:db2://xxx.xxx.xxx.xxx:6789/insdb
ibm-insSystemLogger.console	Turn on console log	"True" or "False" default: *false*

ibm-insSystemLogger.file	Logging output file	`/tmp/sysLog.out` default: *null*
ibm-insTracerLogger.console	Turn on console mask	"True" or "False" default: *false*
ibm-insTracerLogger.file	Trace output file	`/tmp/traceLog.out` default: *null*

- System properties - Universal Notification Dispatcher

setting ID	description	values
ibm-undToBeLogged	List of JLOG record types to log and trace	TYPE_ERROR, TYPE_MISC_DATA, TYPE_INFO
ibm-undNumberOfDispatcherThread	Number of dispatcher threads UND utilizes to treat incoming requests	10
ibm-undConsoleLogOutput	whether to send und messages to the command console.	"true" or "false" default: *false*
ibm-undLogFileLocation	name of the file to store und messages. If this value is null, und messages will not be logged to a file.	`/tmp/messages.log` default: *null*
ibm-undConsoleTraceOutput	whether to send und trace messages to the command console.	"true" or "false" default: *false*
ibm-undTraceFileLocation	name of the file to store und trace messages. If this value is null, und trace messages will not be logged to a file.	`/tmp/traces.log` default: *null*

- System properties - preferences

setting ID	description	values
ibm-prefMinimumConnectionsPool	Minimum number of connections to open to the SecureWay Directory Managment Tool	5
ibm-prefMaximumConnectionsPool	Maximum number of connections to open to SecureWay Directory Managment Tool	10

ibm-prefMessagesLogFile	Filename where to log the messages	`/tmp/messages.log` default: *null*
ibm-prefTracesLogFile	Filename where to log the traces	`/tmp/traces.log` default: *null*
ibm-prefTracesLogToConsole	Log to console flag for traces logging	"true" or "false" default: *false*
ibm-prefMessagesLogToConsole	Log to console flag for messages logging	"true" or "false" default: *false*

- System properties - adaptors

setting ID	description	values
WirelessGatewayURL	A Wireless Gateway URL to which to connect	for example, http://localhost:13131

Everyplace Intelligent Notification Services - Customization

Several parts of Everyplace Intelligent Notification Services can be customized.

- Subscription triggers: forms, servlets, and trigger handlers
- Content Adapters
- Gateway Adapters
- Transcoders
- Transcoder Style Sheets

Customization is an important part of setting up the Everyplace Intelligent Notification Services. Customers must customize several code components in order to create a notification solution that works for their particular business needs. Everyplace Intelligent Notification Services provides a default or sample implementation of these components.

Related information

- How to customize the sample subscription applications
- How to customize the XMLContentAdapter sample
- How to customize Gateway Adapters
- How to customize Transcoders
- How to customize Transcoder Style Sheets

Everyplace Intelligent Notification Services - subscription customization

- Subscription customization overview
- Subscription development resources
- Subscription sample code installation
- Subscription sample code
- Customize the TriggerHandler for authentication with WebSEAL-Lite or WebSphere Application Server.

Subscription customization overview

Use the Everyplace Intelligent Notification Services Java API to customize triggers.

Everyplace Intelligent Notification Services trigger creation and processing is provided by the following components.

- Subscription JSPs
- Subscription servlets
- Trigger Handlers

Tip: Coordinate changes to the JSPs and servlets. Modifications to the JSPs must be supported by modifications to the servlets and servlet support classes. Likewise, modifications to the servlets and servlet support classes must be reflected in the JSPs.

Where to place compiled code: Place modified and compiled servlet code in a WebSphere Application Server directory. Be sure to include the supporting classes. Then, modify the action URLs in the JSPs to point to the new code. Place the JSPs in a WebSphere Application Server directory also.

Ensure that WebSphere Application Server is properly configured to support the new servlets and JSPs. Refer to the WebSphere Application Server documentation for detailed instructions on how to build and administer a Web application.

Subscription JSPs
Subscription JSPs are the "front-end" of triggers, the components that the subscriber interacts with. The subscriber specifies trigger data -- what type of content the user would like to subscribe to and whether the user wants the content to persist. Click here for a description of how to use the sample subscription JSPs.

Subscription servlets
In the subscription servlets, trigger data and the user's ID are input to the `addTrigger` function of the `triggerManager` class. This function uses the trigger data and user ID to construct a trigger and store it in the database. `addTrigger()` returns a Trigger ID. This Trigger ID is required for

some additional trigger management functions that have not been implemented in this sample.

Note: If you wish to implement trigger administration with the additional triggerManager functions, be sure to capture and store the Trigger ID when it is returned from the `addTrigger()` function. Trigger administration may include giving subscribers the ability to unsubscribe and remove triggers. To implement this type of functionality, modify the subscription servlet to be an un-subscription servlet. Use the `removeTrigger()` function, which takes a Trigger ID as a parameter.

TriggerHandler
TriggerHandlers handle trigger firing events. When content matches a user's filter, the trigger is fired. The Trigger Handler composes a notification message and sends that message to the Universal Notification Dispatcher.

A TriggerHandler uses content options to decide which type of message to compose. This information is originally entered by the subscriber in the subscription JSP. If the content option is "save," the TriggerHandler sets up a string representation of the content in XML format to be stored in the database for later retrieval. It then composes a notification message, containing a URL to access the content, and sends that message to the Universal Notification Dispatcher as a RetentionSpecification. If the content option is "don't save," the TriggerHandler composes a notification message that contains the content and sends that message to the Universal Notification Dispatcher as a NotificationSpecification.

Subscription development resources

Refer to the Everyplace Intelligent Notification Services API Javadoc for details on the classes used to create and process triggers. View the source code below to see how these classes are used. Refer to the subscription sample in the Sample section for instructions on how to run this sample and see this code in action. Refer to the trigger development walkthroughs for step-by-step explanations of the sample code.

- Everyplace Intelligent Notification Services API Javadoc
- Subscription sample description
- Subscription JSP code walkthrough
- Subscription servlet code walkthrough
- Trigger Handler code walkthrough

Sample code installation

The samples are automatically installed with Everyplace Intelligent Notification Services.

- The `.jsp` files are installed in the `usr/IBMEPS/INS/samples/inssamples/web` directory on AIX and the `opt/IBMEPS/INS/samples/inssamples/web` directory on Solaris.
- The `.class` files are installed in the `usr/IBMEPS/INS/samples/inssamples/servlets` directory on AIX and the `opt/IBMEPS/INS/samples/inssamples/servlets` directory on Solaris.
- The `.java` files are installed in the `usr/IBMEPS/INS/samples/` directory on AIX and the `usr/IBMEPS/INS/samples/` directory on Solaris. The source files for each sample are located in the news, stock, and weather subdirectories.

WebSphere Application Server is automatically configured during installation to find the sample JSP and class files in the above listed locations.

Subscription sample code

Subscription JSPs (from the subscription sample)
Click here for a subscription JSP code walkthrough.

- newsSubscriptionForm.jsp
- weatherSubscriptionForm.jsp
- stockSubscriptionForm.jsp

Subscription servlets (from the subscription sample)
Click here for a subscription servlet code walkthrough.

- newsSubscriptionServlet.java
- weatherSubscriptionServlet.java
- stockSubscriptionServlet.java

Subscription Data classes
A subscription data class holds subscription information such as topic, subject, trigger option, content option, and trigger ID. These classes hold subscription data and allow other classes to get and set the data.

- newsSubscriptionData.java
- weatherSubscriptionData.java
- stockSubscriptionData.java

Subscriptions classes
A subscriptions class contains a vector of subscriptionData instances. These classes allow other classes to get, set, and remove subscriptionData instances from the vector.

- newsSubscriptions.java
- weatherSubscriptions.java
- stockSubscriptions.java

Trigger Handlers (from the subscription sample)
Click here for a Trigger Handler code walkthrough.

- newsHandler.java
- weatherHandler.java
- stockHandler.java

Servlet for retrieving persistent content
This Servlet retrieves persistently stored content and transcodes the retrieved data into HTML format.

- contTransServlet.java

Customize the TriggerHandler for authentication with WebSEAL-Lite or WebSphere Application Server.

The TriggerHandler sends notification messages to the Universal Notification Dispatcher as either a NotificationSpecification or a RetentionSpecification, depending on the user's selected content option. If the content option is "save," the TriggerHandler sends out a RetentionSpecification, containing a URL from which the user can retrieve the persistently stored content.

When authentication is enabled, this URL must route the user through the authentication server. Customize the TriggerHandler code that composes the URL for the RetentionSpecification to route the user properly through authentication. The current sample implementation simply gives the servlet host and content retrieval sample servlet as the RetentionSpecification URL.

The content retrieval sample servlet is *ContTransServlet.class*.The sample code for the sample TriggerHandlers and the ContTransServlet are listed above.

Related information

- Subscription JSP code walkthrough
- Subscription servlet code walkthrough
- Trigger Handler code walkthrough
- Everyplace Intelligent Notification Services API Javadoc
- Notification Markup Language specification
- How to run the subscription sample

Everyplace Intelligent Notification Services - content adapter customization

Content adapter - connecting to a content provider

The Content Adapter publishes data from a content provider to the iQueue Server. In the case of the XMLContentAdapter sample, the content provider is static, composed of an XML file. An actual subscription solution would most likely use data from a live Internet feed. The Content Adapter then transforms the data into an iQueue-readable format.

The data retrieval method depends upon the type of content provider the subscription system utilizes. Various types of content providers are available from vendors. Content providers vary greatly from one to the other, and Content Adapter code must be customized specifically for each. However, here are descriptions of two possible methods for retrieving data from the feeds: file parsing and port listening.

File parsing
A content provider may, upon prompting, store current data in a file. For this type of feed, you may wish to set up an automatic data prompting program that will periodically prompt the feed to store current data in a file. For this type of content provider, retrieve the data by parsing the file. To implement this form of data retrieval, write a new parsing method in the XMLContentAdapter class that makes sense for the content provider file format. You may wish to rename the method and the class to conceptually match the content provider. Then run the application using its new name and passing it the directory location of the data files. Refer to the "Running the sample" section above for more details on how to run the content adapter and which command line arguments to pass to it.

Port listening
A content provider may, instead of periodically storing data in a file, periodically send the data directly to the ContentAdapter. In this case, the ContentAdapter will need a PortListener class to listen to the port for data coming down the feed. To implement this form of data retrieval, eliminate the parsing method. Write a class that will listen to the data port and deserialize, parse, and format the data it receives. Then replace the file parsing code with code to add this new listener and handle the data from it.

ContentBody limitations

ContentBody is the class that holds a string representation of XML-formatted content published to the iQueue Server by a ContentAdapter. This string representation of the content must be less than or equal to 64 KB in length. To publish content greater then 64 KB in length requires breaking the content into 64 KB segments.

Installing samples

The samples are automatically installed with Everyplace Intelligent Notification Services.

- The `.class` files are installed in the `usr/IBMEPS/INS/samples/inssamples/servlets` directory on AIX and the `opt/IBMEPS/INS/samples/inssamples/servlets` directory on Solaris.
- The `.java` files are installed in the `usr/IBMEPS/INS/samples/` directory on AIX and the `usr/IBMEPS/INS/samples/` directory on Solaris. The source files for each sample are located in the news, stock, and weather subdirectories.

WebSphere Application Server is automatically configured during installation to find the sample JSP and class files in the above listed locations.

ContentAdapter sample code

(from the subscription sample)

- XMLContentAdapter.java

Note: The sample content adapter retrieves content from sample, static XML pages. In order to publish meaningful content for users, subscribe to a content provider and download the content provider's client application. You may also develop a content retrieval application to extract content information from Web pages.

Related information

- How to run the ContentAdapter.

Everyplace Intelligent Notification Services - Gateway Adapter customizable code

Gateway Adapters connect the Universal Notification Dispatcher to the Everyplace Wireless Gateway and any other gateways being used. Each recipient device type requires a gateway adapter. For more information on Gateway Adapters, refer to the Gateway Adapters part of the Components section.

- Customize Gateway Adapters
- IBM Messaging Services and Push toolkit
- Gateway Adapter and Transcoder properties in SecureWay Directory

Customize Gateway Adapters

Several of the gateway adapters are designed to interface with the Enterprise Wireless Gateway. It is recommended that the Enterprise Wireless Gateway be used, although the adapters may be modified to work with other gateways.

View the Gateway Adapter source code below.

- Gateway interface
- PAPUser - superclass of sample gateway adapters
- WAP Gateway Adapter
- Sametime/IM Gateway Adapter
- Short Messaging Service (SMS) Gateway Adapter
- SMTP E-mail Gateway Adapter
- E-mail Push Gateway Adapter

To customize the Gateway Adapters,

1. Modify or extend the classes in the source code listed above to work with the alternative gateway. You may also create new Gateway Adapter classes by implementing the Gateway interface.
2. Give the classes new names to distinguish them from the original code.
3. Create a new `.jar` file with the new or modified classes, for example, `adapter.jar`.
4. Place the new `adapter.jar` file in the `lib` directory where the `ins.jar` file is located.
   `/usr/IBMEPS/INS/lib/adapter.jar` on AIX
   `/opt/IBMEPS/INS/lib/adapter.jar` on Solaris
5. Modify the startup script for Universal Notification Dispatcher to include the `adapter.jar` file in the CLASSPATH.
   `/usr/IBMEPS/INS/bin/startUND` on AIX
   `/opt/IBMEPS/INS/bin/startUND` on Solaris
   Note: Ensure that `adapter.jar` is before `ins.jar` in the CLASSPATH in order for the classes in `adapter.jar` to override the corresponding original classes in `ins.jar`. For example, set the CLASSPATH to
   ```
   CLASSPATH=/usr/IBMEPS/INS:/usr/IBMEPS/INS/bin:
   /usr/IBMEPS/INS/lib/adapter.jar:/usr/IBMEPS/INS/lib/ins.jar
   ```
6. Modify the package name settings in SecureWay Directory to reflect the new class or package names. These settings will tell the Universal Notification Dispatcher to instantiate adapters using the new, customized classes. Refer to the section on "Gateway Adapter and Transcoder configuration in SecureWay Directory" for more details on how to modify these settings in SecureWay Directory.
7. Restart the Universal Notification Dispatcher with the modified startup script. Click here for more information

on starting and stopping Intelligent Notification Services.

IBM Messaging Services and Push toolkit

The Gateway Adapter sample code implements WAP, SMS, and E-mail Push Adapters based on the IBM Messaging Services and Push toolkit. The IBM Messaging Services and Push toolkit interacts with Everyplace Wireless Gateway. The Gateway Adapter superclass, PAPUser, handles common interactions with Everyplace Wireless Gateway using the `com.ibm.wireless.push` package.

The toolkit may be downloaded from http://www-3.ibm.com/pvc/tech/downloads.shtml

Gateway Adapter and Transcoder configuration in SecureWay Directory

Gateway Adapters and Transcoders are registered in SecureWay. Based upon the name and location properties stored in SecureWay Directory, the Notification Dispatcher instantiates the appropriate adapter for the recipient device. SecureWay Directory stores the Adapter and Transcoder names and locations. View or modify the Adapter and Transcoder properties in SecureWay using the Directory Management Tool.

1. To edit these properties using the Directory Management Tool, select the Browse Tree option. A SecureWay tree similar to the following one will appear.

```
dc=raleigh,dc=ibm,dc=com
      sys=SDP
              .....
              sys=und
                     cid=common
                             settingID=...
                             .....
                             settingID=ibm-undDeviceAdaptorLink
                             settingID=ibm-undDeviceTranscoderLink
                             settingID=ibm-undStyleSheetLink
                             .....
              sys=ins
              sys=otherComponents
```

2. Right-click on one of the appropriate ibm-und settingIDs. An "Edit an LDAP entry" dialog will appear. Go to the Attributes panel and view or edit the cesProperty. A cesProperty will look like: `SMS_TC=com.ibm.pvc.we.ins.und.server.adaptors.transcoder.SMSTranscoder`, where `SMS_TC` is the transcoder entry name and `com.ibm.pvc.we.ins.und.server.adaptors.transcoder.SMSTranscoder` is the transcoder class.

3. Modify the location listed above to reflect the new class and package name of the Transcoder, Gateway Adapter, or style sheet. For example, to replace the sample SMS transcoder with another transcoder, replace
   `SMS_TC=com.ibm.pvc.we.ins.und.server.adaptors.transcoder.SMSTranscoder`
   with
   `SMS_TC=newpackage.newTranscoder`

Related information

- Gateway Adapter description
- How to start and stop the Everyplace Intelligent Notification Services

Everyplace Intelligent Notification Services - Transcoder customizable code

Each Gateway Adapter has its own Transcoder. Each Transcoder uses a Transcoder Style Sheet if one is available. Otherwise, the Transcoder will use its default formatting method. Modify the sample Transcoder stylesheet for a particular device to customize the display formatting of messages for that device.

- Transcoder customization overview
- Gateway Adapter and Transcoder properties in SecureWay Directory

Transcoder customization overview

The Transcoders provided should cover most scenarios. However if different or additional function is required, the Transcoders may be modified, or the Generic Transcoder extended.

View the Transcoder source code below.

- Transcoder interface
- TcFactory (factory class that is used to create instances of objects implementing the Transcoder interface.
- Generic Transcoder (super class for sample transcoders)
- WAP Transcoder
- Sametime/IM Transcoder
- Short Messaging Service (SMS) Transcoder
- SMTP E-mail Transcoder
- E-mail Push Transcoder

To customize the Transcoders,

1. Modify or extend the classes in the source code listed above to work with the alternative gateway. You may also create new Transcoder classes by implementing the Transcoder interface.

   The Transcoder code for each device implements a default message formatting for that device. This default message formatting is used in the case when a Transcoder Stylesheet is not available. Therefore, to customize the message formatting, modify or override the formatting function and ensure that this function will be used by making the Transcoder stylesheets unavailable.

   The Transcoder formatting function transforms messages from Notification Markup Language (Notification ML) into device-readable format. To better understand how this formatting function works, refer to the source code listed above and to the Notification ML part of the Samples section of this documentation.

2. Give the classes new names to distinguish them from the original code.

3. Create a new `.jar` file with the new or modified classes, for example, `transcoder.jar`.

4. Place the new `transcoder.jar` file in the `lib` directory where the `ins.jar` file is located.
   `/usr/IBMEPS/INS/lib/transcoder.jar` on AIX
   `/opt/IBMEPS/INS/lib/transcoder.jar` on Solaris

5. Modify the startup script for Universal Notification Dispatcher to include the `transcoder.jar` file in the CLASSPATH.
   `/usr/IBMEPS/INS/bin/startUND` on AIX
   `/opt/IBMEPS/INS/bin/startUND` on Solaris
   Note: Ensure that `transcoder.jar` is before `ins.jar` in the CLASSPATH in order for the classes in `transcoder.jar` to override the corresponding original classes in `ins.jar`. For example, set the

CLASSPATH to
`CLASSPATH=/usr/IBMEPS/INS:/usr/IBMEPS/INS/bin:`
`/usr/IBMEPS/INS/lib/transcoder.jar:/usr/IBMEPS/INS/lib/ins.jar`

6. Modify the package name settings in SecureWay Directory to reflect the new class or package names. These settings will tell the Universal Notification Dispatcher to instantiate transcoders using the new, customized classes. Refer to the section on "Gateway Adapter and Transcoder configuration in SecureWay Directory" for more details on how to modify these settings in SecureWay Directory.

7. Restart the Universal Notification Dispatcher with the modified startup script. Refer to the Administer section for more information on starting and stopping Intelligent Notification Services.

Gateway Adapter and Transcoder configuration in SecureWay Directory

Gateway Adapters and Transcoders are registered in SecureWay Directory. Based upon the name and location properties stored in SecureWay Directory, the Universal Notification Dispatcher instantiates the appropriate adapter for the recipient device. SecureWay Directory stores the Adapter and Transcoder names and locations. View or modify the Adapter and Transcoder properties in SecureWay Directory using the Directory Management Tool.

1. To edit these properties using the Directory Management Tool, select the Browse Tree option. A SecureWay tree similar to the following one will appear.

```
dc=raleigh,dc=ibm,dc=com
      sys=SDP
              .....
              sys=und
                      cid=common
                              settingID=...
                              .....
                              settingID=ibm-undDeviceAdaptorLink
                              settingID=ibm-undDeviceTranscoderLink
                              settingID=ibm-undStyleSheetLink
                              .....
              sys=ins
              sys=otherComponents
```

2. Right-click on one of the appropriate ibm-und settingIDs. An "Edit an LDAP entry" dialog will appear. Go to the Attributes panel and view or edit the cesProperty. A cesProperty will look like: `SMS_TC=com.ibm.pvc.we.ins.und.server.adaptors.transcoder.SMSTranscoder`, where `SMS_TC` is the transcoder entry name and `com.ibm.pvc.we.ins.und.server.adaptors.transcoder.SMSTranscoder` is the transcoder class.

3. Modify the location listed above to reflect the new class and package name of the Transcoder, Gateway Adapter, or style sheet. For example, to replace the sample SMS transcoder with another transcoder, replace
`SMS_TC=com.ibm.pvc.we.ins.und.server.adaptors.transcoder.SMSTranscoder`
with
`SMS_TC=newpackage.newTranscoder`

The default transcoder keys are as follows:

```
WAP_TC= com.ibm.pvc.we.ins.und.server.adaptors.transcoder.WAPTranscoder
IMSAMETIME1V5_TC=com.ibm.pvc.we.ins.und.server.adaptors.transcoder.IMSTTranscoder
```

```
SMS_TC=com.ibm.pvc.we.ins.und.server.adaptors.transcoder.SMSTranscoder
EMAIL_TC=com.ibm.pvc.we.ins.und.server.adaptors.transcoder.EMAILTranscoder
MAILPUSH_TC=com.ibm.pvc.we.ins.und.server.adaptors.transcoder.MailPushTranscoder
```

Related information

- How to start and stop Everyplace Intelligent Notification Services
- Notification Markup Language specification

Everyplace Intelligent Notification Services - transcoder stylesheets

Each gateway adapter has a transcoder that formats notification messages for the target device. Transcoders use stylesheets for formatting if one is available. Modify the sample Transcoder stylesheet for a particular device to customize the formatting of messages for that device.

Everyplace Intelligent Notification Services sample transcoders have a required naming convention for loading stylesheets. The sample transcoders use the transcoder class name and append _SS to construct a key for loading stylesheets. For example,

`IMSTTranscoder_SS=/usr/IBMEPS/INS/samples/adaptors/IMSTTranscoder_SS.xsl`.

Note: This convention must be used with the sample transcoder. The sample transcoder uses the transcoder class name and appends _SS to construct a key to load the stylesheets.

The Everyplace Intelligent Notification Services stylesheets are part of the Transcoder package since transcoders transform the NotificationML into the device specific markup language. Although style sheets are not required, they do enhance formatting and improve performance.

Device types

- Wireless Application Protocol (WAP)
- Short Message Service (SMS)
- E-mail
- Instant Messaging Service (IMS)

Style sheets (customizable code)

- *WAPTranscoder_SS.xsl*
- *SMSTranscoder_SS.xsl*
- *EMAILTranscoder_SS.xsl*
- *IMSTranscoder_SS.xsl*

Notification Markup Language

These style sheets are used to convert messages from Notification ML format into device-specific, displayable format. Click here for more information about the Notification Markup Language.

Related information

- *WAPTranscoder_SS.xsl*
- *SMSTranscoder_SS.xsl*
- *EMAILTranscoder_SS.xsl*
- *IMSTranscoder_SS.xsl*
- Notification Markup Language specification

Notification ML specification

Notification Markup Language (Notification ML)

Notification ML is the markup language used to format a notification message. Notification ML is a form of XML. The supported tags are described in the table below.

Sample Notification Message:

```
<message>
        <to>Joe_employee@realm1.company.com</to>
        <from>CompanyX_EINS@realm1.company.com</from>
        <subject>CompanyX Stock</subject>
        <text>CX = 110</text>
        <url>http://Svcs_persistant_storage/stock38521.jsp</url>

</message>
```

Document Type Definition (DTD) for Notification ML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT message (to, from, subject, text, url*)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

Supported Notification ML tags

tag	description
<message>...</message>	notification message
<to>...</to>	userid of message recipient
<from>...</from>	userid of message sender

<subject>...</subject>	subject of message
<text>...</text>	text message body
<url>...</url>	link to a stored message

Everyplace Intelligent Notification Services API Javadoc

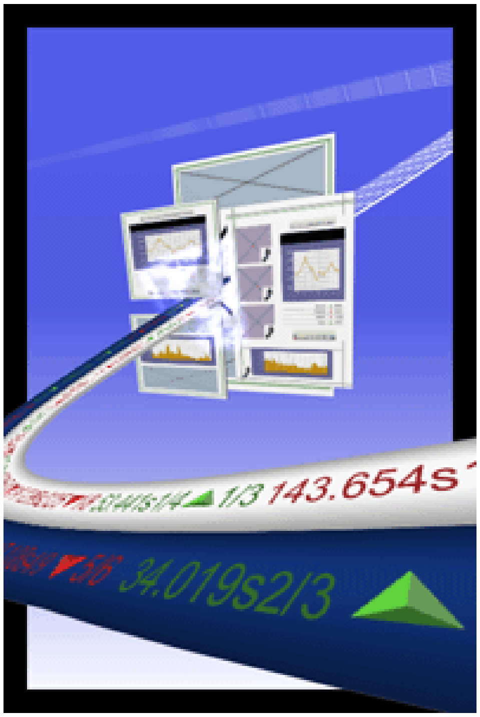
The Everyplace Intelligent Notification Services has several Application Programming Interfaces (APIs) for use in customizing components. Refer to this documentation for descriptions of key classes and lists and descriptions of the fields and methods for those classes. This Javadoc can be used as a reference to aid in the customization of subscription JSPs, subscription servlets, trigger handlers, and Universal Notification Dispatcher calls to `sendMessage()`.

- Click here to view the Everyplace Intelligent Notification Services API Javadoc. (This will bring up the Javadoc in a Separate browser window.)

Related information

- Everyplace Intelligent Notification Services API Javadoc

Everyplace Intelligent Notification Services - Subscription Sample



Subscription overview

This sample application allows users to register a notification subscription for stock quotes, weather, or news. It demonstrates the end-to-end life cycle of a user subscription. It also provides samples of the code that must be customized to get a subscription solution working for your company -- from the end-user (subscriber) JSPs to the customizable inner workings of iQueue server such as Trigger Handlers and Content Adapters.

We provide three subscription samples.

- Stock Quote Subscription
- Weather Subscription
- News Subscription

Follow the "Run this sample" instructions below to see how to connect to a content feed and what the end-user JSPs look like. Read through the description below to understand how the sample works and which parts of the sample must be customized to enable one's own subscription solution.

Installing samples

The samples are automatically installed with Everyplace Intelligent Notification Services.

- The `.jsp` files are installed in the `usr/IBMEPS/INS/samples/inssample/web` directory on AIX and the `opt/IBMEPS/INS/samples/inssample/web` directory on Solaris.
- The `.class` files are installed in the `usr/IBMEPS/INS/samples/inssample/servlets` directory on AIX and the `opt/IBMEPS/INS/samples/inssample/servlets` directory on Solaris.
- The `.java` files are installed in the `usr/IBMEPS/INS/samples/` directory on AIX and the

`usr/IBMEPS/INS/samples/` directory on Solaris. The source files for each sample are located in the news, stock, and weather subdirectories.

WebSphere Application Server is automatically configured during installation to find the sample JSP and class files in the above listed locations.

Run this sample

1. Prerequisites:
   Ensure that the following components are installed and running.
   - iQueue Server
   - WebSphere Application Server
   - Universal Notification Dispatcher

   Click here for instructions on how to start and stop the iQueue Server and Universal Notification Dispatcher.

2. Point a Web browser towards the front-end JSPs. The sample JSPs have automatically been installed in a WebSphere Application Server directory. The URLs are
   - `<hostname>:<port#>/inssample/NewsSubscriptionForm.jsp`
   - `<hostname>:<port#>/inssample/StockSubscriptionForm.jsp`
   - `<hostname>:<port#>/inssample/WeatherSubscriptionForm.jsp`

   Note: If Web server security has not been turned on, you must provide a userid when calling the JSPs. The userid must contain the domain. For example,
   - `<hostname>:<port number>/inssample/NewsSubscriptionForm.jsp?userid=youruserid@host.domain.com`

3. Register a subscription using the JSP forms.

4. Run the *contentfeeder* script. The *contentfeeder* script sets the CLASSPATH and starts the *XMLContentAdapter* application, using the sample content feeds.

   Type the following commands.
   - `/usr/IBMEPS/INS/samples/contentfeeder.sh <topic> <sample data directory>` on AIX
   - `/opt/IBMEPS/INS/samples/contentfeeder.sh <topic> <sample data directory>` on Solaris

   `<topic>` is `news`, `weather`, or `stocks` and `<sample data directory>` is the location of the static sample content providers, usually the install location of the Everyplace Intelligent Notification Services samples.

   The XMLContentAdapter will parse all of the XML files in the directory that you specify on the command line. The `<topic>` command line argument specifies the topic for the content source that you are providing. This must match the topic specified on the subscription for a subscription to match.

How this sample works

This sample uses the Everyplace Intelligent Notification Server application programming interface (API) to create an application that enables the user to request notification subscriptions. Subscription requests submit triggers to the iQueue server. The triggers are associated with a content source as well as with a firing condition. When the firing condition becomes valid, one or more actions are taken on behalf of the user. These actions are based on a trigger handler which is supplied as part of the sample application.

A Content Adapter listens to various data streams such as a news wire service. It interprets the data and listens for keywords. When keywords are detected, the Content Adapter publishes the content in the form of name-value pairs to the iQueue server which filters the content.

When the Trigger Manager on the iQueue Server detects a match, it executes the appropriate trigger handler. The trigger handler submits the notification specification in NotificationML to the Universal Notification Dispatcher (UND) indirectly through the iQueue Server. The UND is responsible for dispatching messages to subscribers based upon their selected preferences.

Subscription data and persistent messages are stored on the iQueue Server.

Reasons for message delivery failure: Notification messages may fail to be sent for a variety of reasons. The following list gives examples of possible reasons for delivery failure.

- The recipient is offline for instant messaging.
- No devices were specified for the recipient.
- The message sender is specified as part of a group that does not have access to the recipient's devices.

Refer to the NotificationService class documentation in the API Javadoc for a full explanation of sendMessage() function return values. The return values specify reasons for delivery failure.

If the trigger notification preference is set to "Always," delivery attempts will continue to be made, despite previous failed delivery. If the trigger notification preference is set to "Once," no additional delivery attempts will be made.

How to use the JSPs - subscriber interactions

Click here for more information about how to customize the sample subscription JSPs.

Notification preferences selection
For each subscription sample JSP, you can select notification preferences. The user has two frequency selections, once and always. Once means the user is notified only the first time when the criterion is met. Always means the user is notified every time the criterion is met.

Save option
In each subscription sample, the user may also select a save option. When the save option is selected, the content is saved in the iQueue server database. A URL for accessing the content is then sent in the notification message to the end user.

Refresh button
In each subscription sample, the user may click on the Refresh button to remove any entries in the subscription table that have been marked to be deleted.

Submit request button
When the user selects the Submit button, the sample application registers the subscription with Trigger Manager.

Retrieve persistent messages
If the save option is selected, then large content like news articles can be stored on a database by the iQueue Server. The user is sent the URL of the *Retrieve persistent messages* servlet in the message.The user can enter the URL in a Web browser. The servlet retrieves the news article from the database and delivers it to the user's Web browser. The sample servlet for retrieving persistent content is called *ContTransServlet.java*.

Content Adapter - publishing data from a content provider

Click here for more information on how to set up and customize a content adapter.

The Content Adapter publishes data from a content provider to the iQueue Server. In the case of this sample, the content provider is a static one composed of an XML file. An actual subscription solution would most likely use data from a live Internet feed. The Content Adapter transforms the data into an iQueue-readable format.

ContentBody limitations
ContentBody is the class that holds a string representation of XML-formatted content published to the iQueue Server by a ContentAdapter. This string representation of the content must be less than or equal to 64 KB in length. To publish content greater then 64 KB in length requires breaking the content into 64 KB segments.

Trigger Handler - determining how to notify the subscriber

Click here for more information on how to customize a Trigger Handler.

Trigger Handlers handle trigger firing events. When content matches a user's filter, the trigger is fired. The Trigger Handler composes a notification message and sends that message to the Universal Notification Dispatcher.

A Trigger Handler uses content options to decide which type of message to compose. This information is originally entered by the end user on the subscription JSP. If the content option is "save," the Trigger Handler sets up a string representation of the XML formatted content to be stored in the persistent storage for later retrieval. It then composes a notification message, containing a URL to access the content, and sends that message to the Universal Notification Dispatcher as a Retention Specification. If the content option is "don't save," the Trigger Handler composes a notification message that contains the content and sends that message to the Universal Notification Dispatcher as a Notification Specification.

Related information

- How to start and stop the iQueue Server and Universal Notification Dispatcher
- How to customize the subscription JSPs, servlets, and Trigger Handlers of the subscription sample applications
- How to customize a ContentAdapter
- Steps for getting a notification system up and running

Everyplace Intelligent Notification Services - Universal Notification Dispatcher API sample

INSTest is a sample application that demonstrates how to use the Notification Dispatcher API. This sample application is in a file called *INSTest.java*. It uses the `sendMessage(String notificationML, DeliveryOptions senderPrefs)` method of the NotificationService class to compose the message in Notification ML format and specify delivery options -- message priority, devices, and which devices to send to.

- View INSTest code. (This link will bring up the code in a separate browser window.)
- Click here to view the Notification ML specifications.
- Click here to view the Everyplace Intelligent Notification Services API Javadoc. (This link will bring up the code in a separate browser window.)

Reasons for message delivery failure: Notification messages may fail to be sent for a variety of reasons. The following list gives examples of possible reasons for delivery failure.

- The recipient is offline for instant messaging.
- No devices were specified for the recipient.
- The message sender is specified as part of a group that does not have access to the recipient's devices.

Refer to the NotificationService class documentation in the API Javadoc for a full explanation of sendMessage() function return values. The return values specify reasons for delivery failure.

If the trigger notification preference is set to "Always," delivery attempts will continue to be made, despite previous failed delivery. If the trigger notification preference is set to "Once," no additional delivery attempts will be made.

Sample code install locations

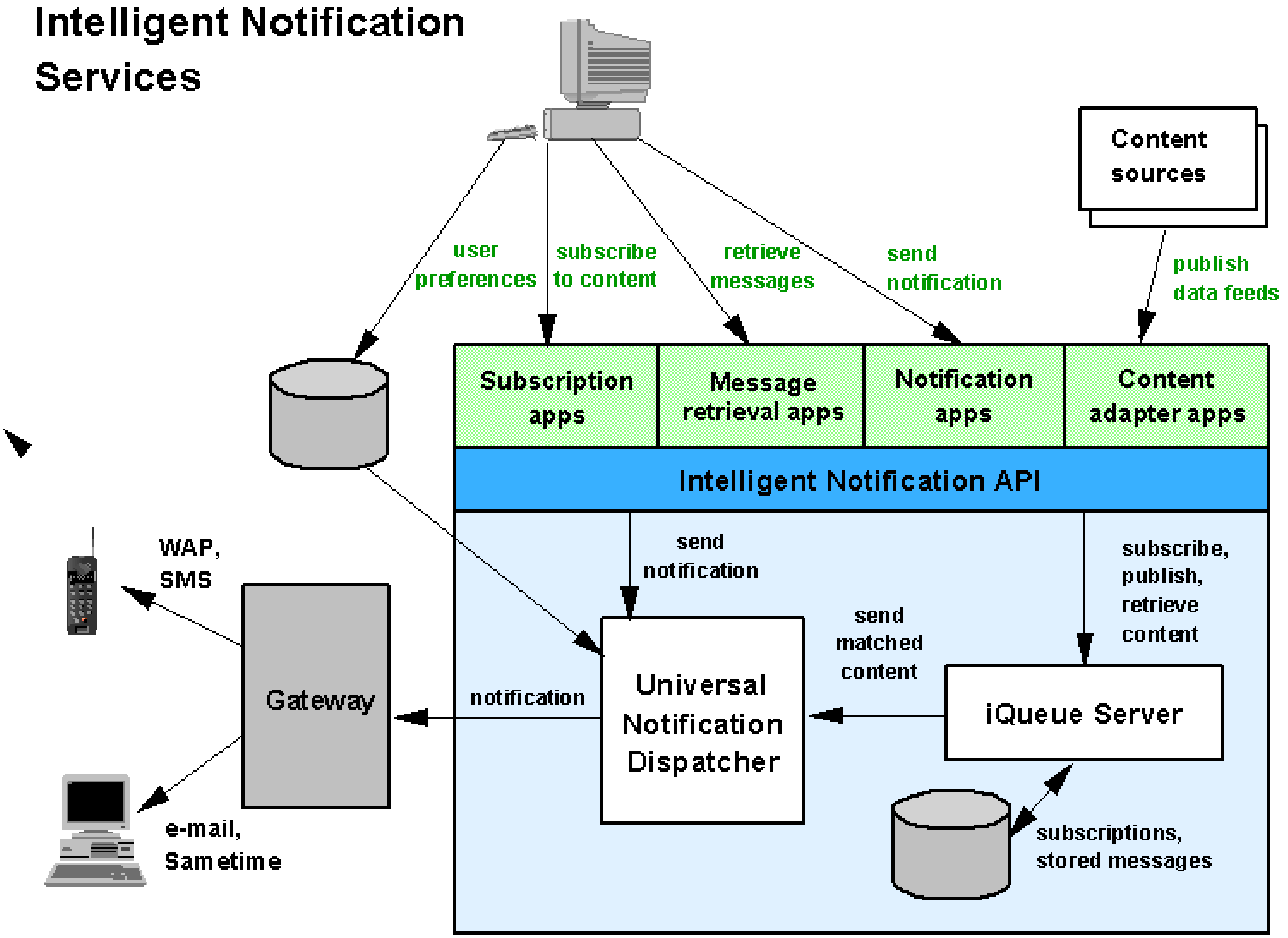
The samples are automatically installed with Everyplace Intelligent Notification Services.

- The `.class` files are installed in the `usr/IBMEPS/INS/samples/inssamples/servlets` directory on AIX and the `opt/IBMEPS/INS/samples/inssamples/servlets` directory on Solaris.
- The `.java` files are installed in the `usr/IBMEPS/INS/samples/` directory on AIX and the `usr/IBMEPS/INS/samples/` directory on Solaris. The source files for each sample are located in the news, stock, and weather subdirectories.

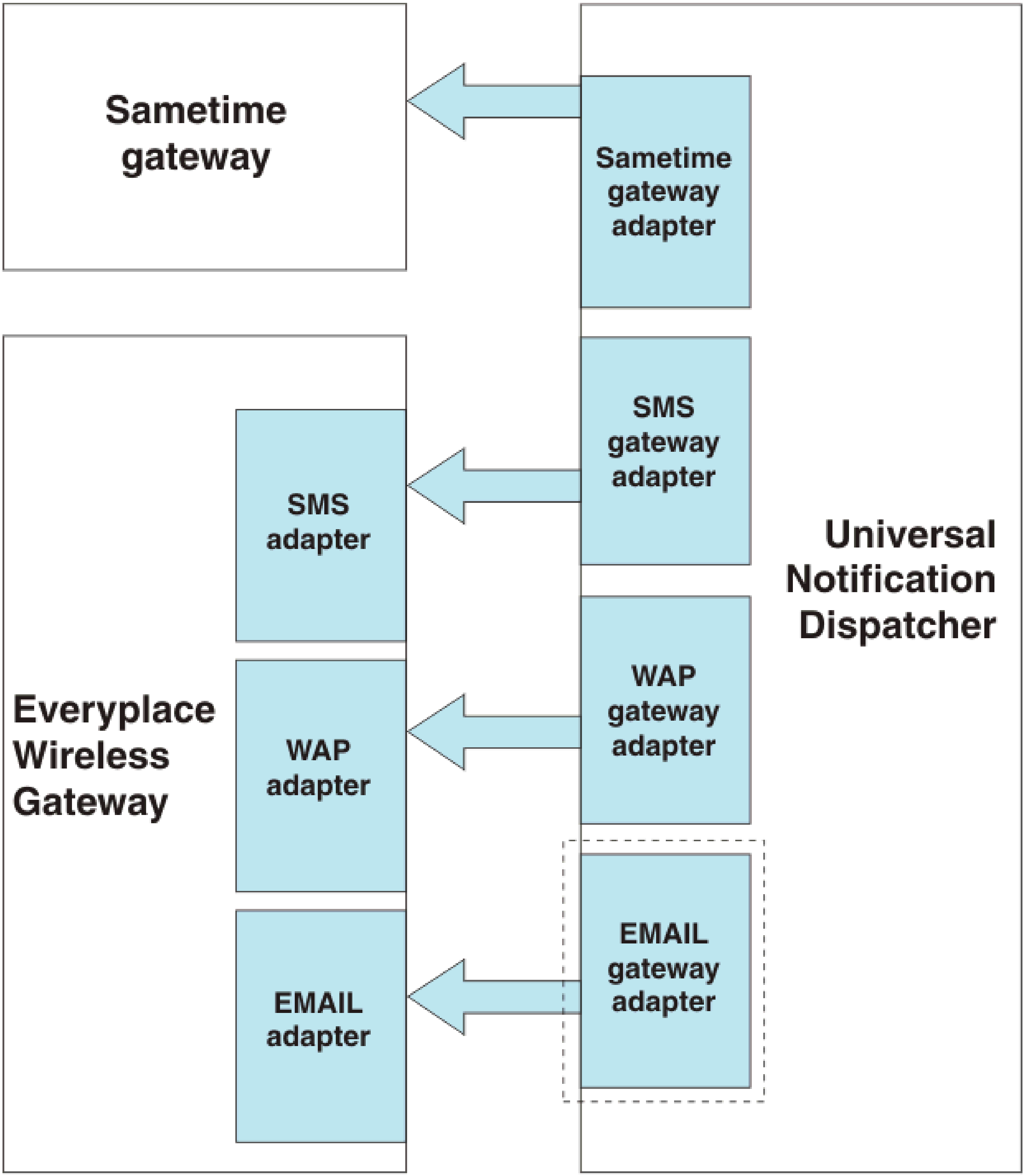
WebSphere Application Server is automatically configured during installation to find the sample JSP and class files in the above listed locations.

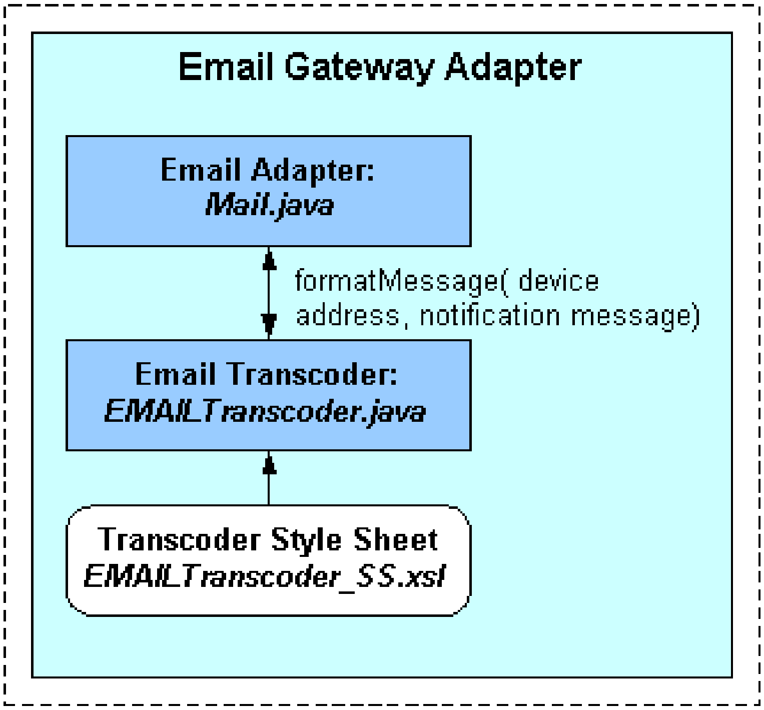
Related information

- View INSTest code.
- View the Notification ML specifications.
- View the Everyplace Intelligent Notification Services API Javadoc.


Intelligent Notification Services
user preferences
subscribe to content
retrieve messages
send notification
Content sources
publish data feeds
Subscription apps
Message retrieval apps
Notification apps
Content adapter apps
Intelligent Notification API
send notification
subscribe, publish, retrieve content
send matched content
WAP, SMS
Gateway
notification
Universal Notification Dispatcher
iQueue Server
e-mail, Sametime
subscriptions, stored messages

Everyplace Intelligent Notification Services - gateway adapter graphical representation




Email Gateway Adapter
Email Adapter:
Mail.java
formatMessage(device address, notification message)
Email Transcoder:
EMAILTranscoder.java
Transcoder Style Sheet
EMAILTranscoder_SS.xsl

Everyplace Intelligent Notification Services - subscription JSP sample code walkthrough

1. Import necessary native packages.
    - `java.io.*`
    - `javax.servlet.*`
    - `javax.servlet.http.*`
    - `javax.servlet.jsp.*`

2. Import necessary subscription classes.
    - `xxxxSubscriptions`
    - `xxxxSubscriptionData`
    - `com.ibm.pvc.we.ins.util.SubscriptionUtility`

    Where `xxxx` is either News, Stock, or Weather.

3. Retrieve the user ID. There are two ways to do this.
    - Retrieve the user ID from the URL if authentication is not implemented.
    - Retrieve the user ID from the header if authentication is implemented.

    Both ways are implemented by the SubscriptionUtility class listed above. The JSP makes a call to SubscriptionUtility.getUser(). The getUser() method checks the iqAuthFlag in the IQ.properties file. The iqAuthFlag indicates whether authentication is implemented or not. If the authentication flag is true, the method retrieves the user ID from the header. If the authentication flag is false, the method retrieves the user ID from the URL.

    Click here for instructions on how to set up authentication.

    Note: If the user ID is retrieved from the URL, the user ID must contain the domain. An example user ID looks like `username@domain.company.com`, and the corresponding example URL looks like `http://localhost:8080/inssample/NewsSubscriptionForm.jsp?userid=username@domain.company.com`.

4. Create a JSP input form. The sample form is HTML with some java code inserted to build the dynamic table and display dynamic fields. The source contains a small section of JavaScript to validate the input fields. The form contains the following input fields.
    - Values that will eventually compose the SQL92 Selector String. There must be at least one Name/Value pair. In the News subscription example, the names are "newssubject" and "newstopic."
    - "trigopt": Notification Option. This String has two possible values.
        - "once"
        - "always"
    - "contopt": Content Storage Option. This String has two possible values.
        - "save"
        - "do not save"
5. The form may also contain the following optional items, as implemented in the News subscription sample.
    - "add" button to add a trigger to the displayed list.
    - "delete" check box to remove a trigger from the displayed list.
    - "submit" button to submit the list of triggers to be added to the trigger manager and stored in the database.

Everyplace Intelligent Notification Services - *SubscriptionUtility.java*

```
/*******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                 *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved             *
 *                                                                             *
 *  Licensed Materials - Property of IBM                                       *
 *                                                                             *
 *  US Government Users Restricted Rights - Use, duplication or disclosure     *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.              *
 *******************************************************************************
*/
package com.ibm.pvc.we.ins.util;

import java.util.*;
import javax.servlet.http.*;
import com.ibm.pvc.we.ins.GeneralConstants;

/* This is a utility class which retrieves properties from
 * the IQ properties file.
 */

public class SubscriptionUtility
{
  private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
  /***********************************************************************/
  /* Name of the properties file containing the host name
  /* and port number of the IQueue server */
  /**********************************************************************/
  private static String IQproperties = "IQ";
  private static ResourceBundle bundle = null;

  /*
   * Method Name: isSecure
   */
   public static boolean isSecure()
   {
      boolean bSecure = false;

      //Load the properties resource file
      bundle = PropertyResourceBundle.getBundle(IQproperties);
      String value = null;
      if (bundle != null)
      {
         value = bundle.getString("iqAuthFlag");
                 //Determine whether authorization services are enabled
         bSecure = Boolean.valueOf(value.trim()).booleanValue();
                 //Convert the string property to a boolean value
      }
      return bSecure;
   }

  /*
   * Method Name: getUser
   */
   public static String getUser(HttpServletRequest request)
   {
      String user = null;

      if (isSecure())
        user = request.getHeader("x-ibm-pvc-user");
      else
        user = request.getParameter("userid");
      return user;
   }

  /*
   * Method Name: getHost
   */

   public static String getHost()
   {
      String host = null;

      //Get the IQueue server host from the properties file
      bundle = PropertyResourceBundle.getBundle(IQproperties);
      if (bundle != null)
         host = bundle.getString("iqSocketClientStub.host");

      return host ;
   }

  /*
   * Method Name: getPort
   */

   public static String getPort()
   {
      String port = null;

      //Get the IQueue server port from the properties file
      bundle = PropertyResourceBundle.getBundle(IQproperties);
      if (bundle != null)
         port = bundle.getString("iqSocketClientStub.port");

      return port;
   }
  /*
   * Method Name: getStyleSheet
   */

   public static String getStyleSheet()
   {
      String stylesheet = null;

      //Get the IQueue server port from the properties file
      bundle = PropertyResourceBundle.getBundle(IQproperties);
      if (bundle != null)
         stylesheet = bundle.getString("insStylesheet.file");

      return stylesheet;
   }

}
```

Everyplace Intelligent Notification Services - *NewsSubscriptionForm.jsp*

```
<!--
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                    *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved                *
 *                                                                                *
 *  Licensed Materials - Property of IBM                                          *
 *                                                                                *
 *  US Government Users Restricted Rights - Use, duplication or disclosure        *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.                 *
-->

<%@ page import="java.io.*, javax.servlet.*, javax.servlet.http.*,
javax.servlet.jsp.*" %>
<%@ page import="NewsSubscriptions, NewsSubscriptionData,
com.ibm.pvc.we.ins.util.SubscriptionUtility" %>
<%@ page info="The News Subscription Form allows a user to subscribe to a news
source and be notified when a certain subject appears in the source." %>


<!-- This is an example of a subscription form for News.
 * This jsp will call the NewsSubscriptionServlet to process the final request.
 -->

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <title>Subscriptions</title>
  <script language=javascript>
      function validate()
      {
         <!-- Validate the input data  -->
         if (document.form1.newssubject.value == "")
         {
           alert('Subject is a required field. Please try again.');
           return false;
         }
         else
           return true;
      }
  </script>
</head>
<body>
  <TABLE border=0 cellPadding=0 cellSpacing=0 width="100%">
    <TR>
      <TD align=left rowSpan=3 vAlign=top width="33%">
        <IMG align=bottom border=0 src="/inssample/yourCo.jpg">
      </TD>
    </TR>
    <%
      String Id;
      Id = SubscriptionUtility.getUser(request);
    %>
    <!-- Add two blank rows since the image takes up three rows in the table
        -->
    <TR></TR>
  <TR></TR>
  <TR>
    <TD align=left vAlign=top><BIG>Welcome <%= Id
        %>!</BIG>
    </TD>
  </TR>
</TABLE>
<HR COLOR=ff8000>
<table border=0 cellPadding=0 cellSpacing=0 width="80%">
  <% if (Id == null)
     { %>
       <tr>
         <font color = "#0000ff">
                 User ID is null.  Contact your administrator.</font>
       </tr>
  <% } %>
  <tr>
    <% if (request.getAttribute("message1") != null)
       { %>
         <font color="#0000ff">
    <%=  request.getAttribute("message1") %>
         </font>
    <% } %>
  </tr>
  <tr>
    <% if (request.getAttribute("message2") != null)
       { %>
         <font color="#0000ff">
    <%=  request.getAttribute("message2") %>
         </font>
    <% } %>
  </tr>
</table>
<BR>
<h2 align=center>News Subscriptions</h2>
<form name=form1 action="/inssample/servlet/news" method=post>
  <OL>
    <LI>Add the following information to my news subscription:</LI>
    <br>
    <br>
    <table border=0 cellPadding=0 cellSpacing=0 width="80%">
      <tr>
        <td>
          <strong>News Source: </strong>
          <FONT COLOR="#ff0000">*</FONT>
          <SELECT name="newstopic">
            <OPTION value="ap.business"> AP:Business </OPTION>
            <OPTION value="newsbytes.news"> Newsbytes Top Stories
                        </OPTION>
            <OPTION value="business_wire.corporate_newsfeed.ft">
              Business Wire: Corporate Industry Press Releases"
                              </OPTION>
            <OPTION value="business_wire.europe.ft">
                        Business Wire: European Press Releases </OPTION>
            <OPTION value="upi.all.ft"> UPI </OPTION>
            <OPTION value="sportsnetwork.news.ft"> Sports Network News
                        </OPTION>
          </SELECT>
        </td>
        <td>
          <strong>Subject: </strong>
        <FONT COLOR="#ff0000">*</FONT>
        <input type="text" name="newssubject" value="" size="20">
      </td>
    </tr>
  </table>
  <br>
  <LI>Indicate your notification and content storage options:</LI>
  <br>
  <ul>
  <li>Notification option: <i>once</i>
      - receive notification only the first time a match occurs;
      <i>always</i> -
      receive notification every time a match occurs.
  <li>Content Storage Option: <i>save</i>
      - Save the message content in storage and provide a link to the content;
      <i>don't save</i>
              - Send the entire message content in the notification.
  </ul>
  <br>
  <table border=0 cellpadding="0" cellspacing="0" width="80%">
    <tr>
      <td>
        <strong>Notification option:  </strong>
        <FONT COLOR="#ff0000">*</FONT>
        <SELECT name="trigopt">
          <OPTION value="once"> once </OPTION>
          <OPTION value="always"> always </OPTION>
          <!-- <OPTION> always and persist </OPTION> -->
        </SELECT>
      </td>
      <td>
        <strong>Content storage option:  </strong>
        <FONT COLOR="#ff0000">*</FONT>
        <SELECT name="contopt">
          <OPTION value="save"> save </OPTION>
          <OPTION value="don't save"> don't save </OPTION>
       </SELECT>
      </td>
    </tr>
  </table>
</OL>
Required fields are indicated by <FONT color="#ff0000">*</FONT>
<br>
<CENTER>
  <input type="submit" name="add" style="border-style: outset"
      value=" Add " onClick="return validate()">
</CENTER>
<% NewsSubscriptions newsSubscriptions =
    (NewsSubscriptions)request.getAttribute("newsSubscriptions"); %>
<TABLE cellspacing="20">
 <TR>
  <TD>
    <TABLE border=2 align="left" cellPadding=5 cellSpacing=0>
      <THEAD>
        <tr>
          <th>Remove </th>
          <th>News Source </th>
          <th>Subject </th>
          <th>Notification </th>
          <th>Content Storage </th>
        </tr>
    </THEAD>
    <TBODY>
    <!-- Description -->
    <TR colSpan=5>
     To remove a subscription, check the Remove
             checkbox referring to the subscription and click Refresh.
    </TR>
    <TR><BR></TR>
    <% try
       {
          //Get the subscriptions currently defined for this user
          NewsSubscriptionData nsd =
                          newsSubscriptions.getSubscription(Id ,0);
                          //Throws an exception if empty
          for (int i = 0; ; )
          {
    %>
    <!-- Unique ID is used for value of the checkbox to identify which
            subscription is being deleted. -->
            <TR>
              <TD><INPUT TYPE="checkbox" NAME="delete"
                                  value="<%= nsd.getId() %>"></TD>
              <TD><%= nsd.getTopic() %></TD>
              <TD><%= nsd.getSubject() %></TD>
              <TD><%= nsd.getTriggerOption() %></TD>
              <TD><%= nsd.getContentOption() %></TD>
            </TR>
    <%
            i++;
            try
            {
              nsd =  newsSubscriptions.getSubscription(Id,i);
            }
            catch (java.lang.ArrayIndexOutOfBoundsException e)
            {break;}
          }
       }
       catch (java.lang.ArrayIndexOutOfBoundsException e) {}
       catch (java.lang.NullPointerException n) {}
                   //This occurs the very first time the page is accessed.
    %>
    </TBODY>
  </TABLE>
</TD>
<TD>
<TABLE>
  <!-- Done for spacing to move the buttons further down on the page -->
  <TR>
    <TD><BR><BR><BR><BR></TD>
  </TR>
  <TR>
    <TD>
      <input type="submit" name="refresh" value="Refresh">
    </TD>
  </TR>
  <TR>
    <TD>
      <input type="submit" name="submit" value="Submit">
    </TD>
  </TR>
</TABLE>
```

```
     </TR>
    </TABLE>
    <% if (!SubscriptionUtility.isSecure())
       { %>
         <INPUT name=userid type="hidden" value="<%= Id %>">
    <% } %>
  </form>
</body>
</html>
```

Everyplace Intelligent Notification Services - *WeatherSubscriptionForm.jsp*

```
<!--
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
-->

<%@ page import="java.io.*, javax.servlet.*, javax.servlet.http.*,
javax.servlet.jsp.*" %>
<%@ page import="WeatherSubscriptions, WeatherSubscriptionData,
com.ibm.pvc.we.ins.util.SubscriptionUtility" %>
<%@ page info="The Weather Subscription Form allows a user to subscribe to
current conditions and/or forecast conditions for a particular city." %>


<!-- This is an example of a subscription form for Weather.
 * This jsp will call the WeatherSubscriptionServlet to process the final request.
 -->

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <title>Subscriptions</title>
  <script language=javascript>
      function validate()
      {
         <!-- Validate the input data -->
         if (document.form1.city.value == "")
         {
           alert('City is a required field.  Please try again.');
           return false;
         }
         else
           if (document.form1.state.value == "")
           {
             alert('State is a required field.  Please try again.');
             return false;
           }
           else
             if (!(document.form1.current.checked) &&
                         !(document.form1.forecast.checked))
             {
               alert('You must select a Weather Report option.
                           Please try again.');
               return false;
             }
             else
               return true;
      }
  </script>
</head>
```

```
<body>
  <TABLE border=0 cellPadding=0 cellSpacing=0 width="100%">
    <TR>
      <TD align=left rowSpan=3 vAlign=top width="33%">
        <IMG align=bottom border=0 src="/inssample/yourCo.jpg">
      </TD>
    </TR>
    <%
      String Id;
      Id = SubscriptionUtility.getUser(request);
    %>
    <!-- Add two blank rows since the image takes up three
        rows in the table -->
    <TR></TR>
    <TR></TR>
    <TR>
      <TD align=left vAlign=top><BIG>Welcome <%=
          Id %>!</BIG>
      </TD>
    </TR>
  </TABLE>
  <HR COLOR=ff8000>
  <table border=0 cellPadding=0 cellSpacing=0 width="80%">
    <% if (Id == null)
       { %>
         <tr>
           <font color = "#0000ff">User ID is null.
                   Contact your administrator.</font>
         </tr>
    <% } %>
    <tr>
      <% if (request.getAttribute("message1") != null)
         { %>
           <font color="#0000ff">
      <%=  request.getAttribute("message1") %>
           </font>
      <% } %>
    </tr>
    <tr>
      <% if (request.getAttribute("message2") != null)
         { %>
           <font color="#0000ff">
      <%=  request.getAttribute("message2") %>
           </font>
      <% } %>
    </tr>
  </table>
  <BR>
  <h2 align=center>Weather Subscriptions</h2>
  <form name=form1 action="/inssample/servlet/weather"
  method=post>
    <OL>
    <LI>Add the following information to my weather subscription:</LI>
    <br>
    <br>
    <table border=0 cellPadding=0 cellSpacing=0 width="80%">
      <tr>
        <td>
          <table border=0 align="left">
            <tr>
              <td colspan=2><strong>Location:</strong>
```

```
                      You must enter both the city and the state.</td>
        </tr>
        <tr>
          <td align="right"><strong>City:
                      </strong><FONT COLOR="#ff0000">*</FONT></td>
          <td align="left"><input type="text" name="city"
                      value="" size="10"></td>
        </tr>
        <tr>
          <td align="right"><strong>State:
                      </strong><FONT COLOR="#ff0000">*</FONT></td>
          <td align="left"><input type="text" name="state"
                      value="" size="2"></td>
        </tr>
      </table>
    </td>
    <td>
      <table border=0 align="right">
        <tr>
          <td><strong>Weather Report
                      (Check all that apply):</strong><FONT
                      COLOR="#ff0000">*</FONT></td>
        </tr>
        <tr>
          <td>
            Current Conditions <input type="checkbox" checked name="current"
                            value="Current Conditions">
          </td>
        </tr>
        <tr>
          <td>
            Forecast <input type="checkbox" checked name="forecast"
                            value="Forecast">
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
<br>
<LI>Indicate your notification and content storage options:</LI>
<br>
<ul>
 <li>Notification option: <i>once</i>
     - receive notification only the first time a match occurs;
     <i>always</i>
     - receive notification every time a match occurs.
 <li>Content Storage Option: <i>save</i>
     - Save the message content in storage and provide a link to the content;
     <i>don't save</i>
             - Send the entire message content in the notification.
</ul>
<br>
<table border=0 cellpadding="0" cellspacing="0" width="80%">
  <tr>
    <td>
      <strong>Notification option:  </strong>
      <FONT COLOR="#ff0000">*</FONT>
      <SELECT name="trigopt">
        <OPTION value="once"> once </OPTION>
        <OPTION value="always"> always </OPTION>
```

```
       <!-- <OPTION> always and persist </OPTION> -->
     </SELECT>
   </td>
   <td>
     <strong>Content storage option:  </strong>
     <FONT COLOR="#ff0000">*</FONT>
     <SELECT name="contopt">
       <OPTION value="save"> save </OPTION>
       <OPTION value="don't save"> don't save </OPTION>
    </SELECT>
   </td>
  </tr>
 </table>
</OL>
Required fields are indicated by <FONT color="#ff0000">*</FONT>
<br>
<CENTER>
  <input type="submit" name="add" style="border-style: outset"
      value=" Add " onClick="return validate()">
</CENTER>
<% WeatherSubscriptions weatherSubscriptions =
    (WeatherSubscriptions)request.getAttribute("weatherSubscriptions"); %>
<TABLE cellspacing="20">
 <TR>
  <TD>
   <TABLE border=2 align="left" cellPadding=5 cellSpacing=0>
    <THEAD>
      <tr>
        <th>Remove </th>
        <th>City </th>
        <th>State </th>
        <th>Weather Report </th>
        <th>Notification </th>
        <th>Content Storage </th>
      </tr>
    </THEAD>
    <TBODY>
    <!-- Description -->
    <TR colSpan=5>
     To remove a subscription, check the Remove checkbox referring
             to the subscription and click Refresh.
    </TR>
    <TR><BR></TR>
    <% try
        {
           //Get the subscriptions currently defined for this user
           WeatherSubscriptionData wsd =
                           weatherSubscriptions.getSubscription(Id ,0);
                           //Throws an exception if empty
           for (int i = 0; ; )
           {
     %>
     <!-- Unique ID is used for value of the checkbox
             to identify which subscription is being deleted. -->
             <TR>
               <TD><INPUT TYPE="checkbox" NAME="delete"
                                   value="<%= wsd.getId() %>"></TD>
               <TD><%= wsd.getCity() %></TD>
               <TD><%= wsd.getState() %></TD>
     <%    if ((Boolean.valueOf(wsd.getCurrent()).booleanValue()) &&
              !(Boolean.valueOf(wsd.getForecast()).booleanValue()))
```

```
                { %>
                    <TD>Current</TD>
          <%    }
                else
                  if ((Boolean.valueOf(wsd.getForecast()).booleanValue()) &&
                     !(Boolean.valueOf(wsd.getCurrent()).booleanValue()))
                  { %>
                    <TD>Forecast</TD>
          <%      }
                  else
                  { %>
                    <TD>Current, Forecast</TD>
          <%      } %>
                    <TD><%= wsd.getTriggerOption() %></TD>
                    <TD><%= wsd.getContentOption() %></TD>
                  </TR>
          <%
                  i++;
                  try
                  {
                    wsd =  weatherSubscriptions.getSubscription(Id,i);
                  }
                  catch (java.lang.ArrayIndexOutOfBoundsException e)
                  {break;}
                }
             }
             catch (java.lang.ArrayIndexOutOfBoundsException e) {}
             catch (java.lang.NullPointerException n) {}
                         //This occurs the very first time the page is accessed.
          %>
         </TBODY>
        </TABLE>
       </TD>
       <TD>
       <TABLE>
         <!-- Done for spacing to move the buttons
                 further down on the page -->
         <TR>
           <TD><BR><BR><BR><BR></TD>
         </TR>
         <TR>
           <TD>
             <input type="submit" name="refresh" value="Refresh">
           </TD>
         </TR>
         <TR>
          <TD>
            <input type="submit" name="submit" value="Submit">
          </TD>
         </TR>
        </TABLE>
       </TR>
      </TABLE>
      <% if (!SubscriptionUtility.isSecure())
         { %>
           <INPUT name=userid type="hidden" value="<%= Id %>">
      <% } %>
  </form>
 </body>
</html>
```

Everyplace Intelligent Notification Services - *StockSubscriptionForm.jsp*

```
<!--
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
-->

<%@ page import="java.io.*, javax.servlet.*, javax.servlet.http.*,
javax.servlet.jsp.*" %>
<%@ page import="StockSubscriptions, StockSubscriptionData,
com.ibm.pvc.we.ins.util.SubscriptionUtility" %>
<%@ page info="The Stock Subscription Form allows a user to subscribe
to a stock and be notified when that stock price hits a certain value." %>


<!-- This is an example of a subscription form for Stock.
 * This jsp will call the StockSubscriptionServlet to process the final request.
-->

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <title>Subscriptions</title>
  <script language=javascript>
      function validate()
      {
         <!-- Validate the input data -->
         if (document.form1.stocksymbol.value == "")
         {
           alert('Stock Symbol is a required field. Please try again.');
           return false;
         }
         else
           if (document.form1.stockprice.value == "")
           {
             alert('Stock Price is a required field. Please try again.');
             return false;
           }
           else
             return true;
      }
  </script>
</head>
<body>
  <TABLE border=0 cellPadding=0 cellSpacing=0 width="100%">
    <TR>
      <TD align=left rowSpan=3 vAlign=top width="33%">
        <IMG align=bottom border=0 src="/inssample/yourCo.jpg">
      </TD>
    </TR>
    <%
      String Id;
    Id = SubscriptionUtility.getUser(request);
  %>
  <!-- Add two blank rows since the image takes up three rows
      in the table -->
  <TR></TR>
  <TR></TR>
  <TR>
    <TD align=left vAlign=top><BIG>Welcome
        <%= Id %>!</BIG>
    </TD>
  </TR>
</TABLE>
<HR COLOR=ff8000>
<table border=0 cellPadding=0 cellSpacing=0 width="80%">
<% if (Id == null)
   { %>
     <tr>
       <font color = "#0000ff">User ID is null.
               Contact your administrator.</font>
     </tr>
<% } %>
  <tr>
    <% if (request.getAttribute("message1") != null)
       { %>
         <font color="#0000ff">
    <%=  request.getAttribute("message1") %>
         </font>
    <% } %>
  </tr>
  <tr>
    <% if (request.getAttribute("message2") != null)
       { %>
         <font color="#0000ff">
    <%=  request.getAttribute("message2") %>
         </font>
    <% } %>
  </tr>
</table>
<BR>
<h2 align=center>Stock Subscriptions</h2>
<form name=form1 action="/inssample/servlet/stock" method=post>
  <OL>
    <LI>
     Add the following information to my stock subscription
         (i.e. Notify me when IBM stock is less than $100):
    </LI>
    <br>
    <br>
    <table border=0 cellPadding=0 cellSpacing=0 width="80%">
     <tr>
       <td>
        <strong>Stock Symbol: </strong>
        <FONT COLOR="#ff0000">*</FONT>
        <input type="text" name="stocksymbol" value="" size="5">
       </td>
     </tr>
     <tr>
      <td>
        <strong>Stock Operation: </strong>
        <FONT COLOR="#ff0000">*</FONT>
        <SELECT name="stockop">
          <OPTION value="="> equal to </OPTION>
          <OPTION value="<"> less than </OPTION>
          <OPTION value=">"> greater than </OPTION>
        </SELECT>
      </td>
     </tr>
     <tr>
      <td>
        <strong>Stock Price: $</strong>
        <FONT COLOR="#ff0000">*</FONT>
        <input type="text" name="stockprice" value="" size="10">
        (i.e. 101.50)
      </td>
     </tr>
   </table>
   <br>
   <LI>Indicate your notification and content storage options:</LI>
   <br>
   <ul>
   <li>Notification option: <i>once</i>
       - receive notification only the first time a match occurs;
       <i>always</i>
       - receive notification every time a match occurs.
   <li>Content Storage Option: <i>save</i>
       - Save the message content in storage and provide a link to the content;
       <i>don't save</i>
               - Send the entire message content in the notification.
   </ul>
   <br>
   <table border=0 cellpadding="0" cellspacing="0" width="80%">
    <tr>
      <td>
        <strong>Notification option:  </strong>
        <FONT COLOR="#ff0000">*</FONT>
        <SELECT name="trigopt">
          <OPTION value="once"> once </OPTION>
          <OPTION value="always"> always </OPTION>
          <!-- <OPTION> always and persist </OPTION> -->
        </SELECT>
      </td>
      <td>
        <strong>Content storage option:  </strong>
        <FONT COLOR="#ff0000">*</FONT>
        <SELECT name="contopt">
          <OPTION value="save"> save </OPTION>
          <OPTION value="don't save"> don't save </OPTION>
        </SELECT>
      </td>
    </tr>
   </table>
  </OL>
  Required fields are indicated by <FONT color="#ff0000">*</FONT>
  <br>
  <CENTER>
    <input type="submit" name="add" style="border-style: outset"
        value=" Add " onClick="return validate()">
  </CENTER>
  <% StockSubscriptions stockSubscriptions =
      (StockSubscriptions)request.getAttribute("stockSubscriptions"); %>
  <TABLE cellspacing="20">
   <TR>
    <TD>
     <TABLE border=2 align="left" cellPadding=5 cellSpacing=0>
```

```
        <THEAD>
         <tr>
          <th>Remove </th>
          <th>Symbol  </th>
          <th>Operation  </th>
          <th>Price  </th>
          <th>Notification </th>
          <th>Content Storage </th>
         </tr>
        </THEAD>
        <TBODY>
         <!-- Description -->
         <TR colSpan=5>
          To remove a subscription, check the Remove checkbox
                  referring to the subscription and click Refresh.
         </TR>
         <TR><BR></TR>
         <% try
             {
                //Get the subscriptions currently defined for this user
                StockSubscriptionData ssd = stockSubscriptions.getSubscription
                                (Id ,0);   //Throws an exception if empty
                for (int i = 0; ; )
                {
          %>
          <!-- Unique ID is used for value of the checkbox to
                  identify which subscription is being deleted. -->
                  <TR>
                    <TD><INPUT TYPE="checkbox"
                                        NAME="delete" value="<%= ssd.getId()
%>"></TD>
                    <TD><%= ssd.getSymbol() %></TD>
                    <TD><%= ssd.getOp() %></TD>
                    <TD><%= ssd.getPrice() %></TD>
                    <TD><%= ssd.getTriggerOption() %></TD>
                    <TD><%= ssd.getContentOption() %></TD>
                  </TR>
          <%
                  i++;
                  try
                  {
                    ssd =  stockSubscriptions.getSubscription(Id,i);
                  }
                  catch (java.lang.ArrayIndexOutOfBoundsException e)
                  {break;}
                }
             }
             catch (java.lang.ArrayIndexOutOfBoundsException e) {}
             catch (java.lang.NullPointerException n) {}
                         //This occurs the very first time the page is accessed.
          %>
        </TBODY>
       </TABLE>
      </TD>
      <TD>
      <TABLE>
       <!-- Done for spacing to move the buttons further down on the page -->
       <TR>
         <TD><BR><BR><BR><BR></TD>
       </TR>
       <TR>
         <TD>
```

```
           <input type="submit" name="refresh" value="Refresh">
         </TD>
       </TR>
       <TR>
         <TD>
           <input type="submit" name="submit" value="Submit">
         </TD>
       </TR>
      </TABLE>
     </TR>
    </TABLE>
    <% if (!SubscriptionUtility.isSecure())
       { %>
         <INPUT name=userid type="hidden" value="<%= Id %>">
    <% } %>
  </form>
</body>
</html>
```

Everyplace Intelligent Notification Services - subscription servlet sample code walkthrough

1. Import the necessary native methods and the Everyplace Intelligent Notification Services package.
   - `import java.io.*;`
   - `import java.util.*;`
   - `import javax.servlet.*;`
   - `import javax.servlet.http.*;`
   - `import com.ibm.pvc.we.ins.*;`

2. Import necessary subscription classes.
   - `xxxxSubscriptions`
   - `xxxxSubscriptionData`
   - `com.ibm.pvc.we.ins.util.SubscriptionUtility`

   Where `xxxx` is either News, Stock, or Weather.

3. Create public class extending HttpServlet.

```
public class SubscriptionServlet extends HttpServlet
{
```

4. Create a String object to identify the ContentSource.

```
   static final String sourceName = "source";
```

5. Override the HttpServlet `doPost()` method. Generally, this method is used to retrieve any data that was entered on an HTML form by the user. This is done using the HttpServletRequest object which is passed into this method. The sample servlets use this method as a router to route the request to the appropriate method based on the type of the request (add, refresh or submit).

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
 {
    userid = SubscriptionUtility.getUser(request);

    if (request.getParameter("add") != null)
        //Add button was pressed
      newsAddData(request, response);
    else
      if (request.getParameter("refresh") != null)
              //Refresh button was pressed
        newsDeleteData(request, response);
      else
              //Submit button was pressed
        newsSubscription(request, response);
 }
```

6. Create methods to add and delete subscriptions from the table.
   1. `xxxAddData(HttpServletRequest request, HttpServletResponse response)`
   2. `xxxDeleteData(HttpServletRequest request, HttpServletResponse response)`

   Note: `xxx` is either News, Stock, or Weather.

   Use the "getParameter" method of the HttpServletRequest object to create a series of String objects with values equal to parameters passed by the Form.

```
final String userid = request.getParameter("userid");
final String names = request.getParameter("name");
final String values = request.getParameter("value");
```

7. Construct a method to register the subscriptions (triggers) with the iQueue Server.

1. Create a Vector object for each String parameter to be used to construct the SQL92 selector String.

   ```
   Vector topics = new Vector();
   Vector subjects = new Vector();
   Vector contentOptions = new Vector();
   Vector triggerOptions = new Vector();
   ```
2. Declare a String object to contain the concatenated SQL92 Name/Value pairs.

   ```
       String sqlsel;
   ```
3. Set the content type to "text/html" using the "setContentType" method of the HttpServletResponse object.

   ```
       response.setContentType("text/html");
   ```
4. Retrieve the subscription data from `xxxSubscriptionData`, where `xxx` is News, Stock, or Weather. Get the first SubscriptionData item from the DataList, retrieve the topics, subjects, trigger options, and content options from the SubscriptionData item, then iterate through the rest of the SubscriptionData items in the DataList, retrieving the same data from each.
5. Connect to Everyplace Intelligent Notification Services using a Trigger Manager.

   ```
       try
       {
   ```
   1. Instantiate a TriggerManager object.

      ```
                      TriggerManager manager = new SocketClientStub(IQueuehost,
      IQueueport);
      ```
   2. Instantiate a ContentSource object.

      ```
                      ContentSource source =
                      SimpleContentSource.newSimpleContentSource(topicNews);
      ```
6. Create trigger and add the trigger to the trigger manager.
   1. Declare a ContentFilter object.

      ```
                      ContentFilter filter;
      ```
   2. Declare a TriggerHandler object.

      ```
                      ThisHandler triggerHand;
      ```
   3. Declare a TriggerID object.

      ```
                      TriggerID tid;
      ```
   4. Create a loop to parse data from the Vector objects, build triggers and add them to the trigger manager.
      1. Instantiate the TriggerHandler object, parsing subscription data from the vector objects.

         ```
         triggerHand = new ThisHandler(userid, (String)triggerOptions.elementAt(i),
         (String)contentOptions.elementAt(i), URL);
         ```
      2. Use the data to build the SQL selector (sqlsel).
      3. Build the filter from the SQL Selector.

    4. Add the trigger to the trigger manager.

5. Catch IQueueException and send error message to the JSP.

```
catch ( IQueueException iqe )
{
        iqe.printStackTrace(System.out);
        request.setAttribute("newsSubscriptions",newsDataList);
        request.setAttribute("message1",
        "<h3>IQueueException Detected</h3>");
        request.setAttribute("message2",
        "<h4>" + iqe.getMessage() + "</h4>");

        RequestDispatcher dispatcher =
        request.getRequestDispatcher("/NewsSubscriptionForm.jsp");
        dispatcher.forward(request,response);
}
```

6. Catch ConfigurationException and send error message to the JSP.

```
catch ( ConfigurationException e )
{
        e.printStackTrace(System.out);
        request.setAttribute("newsSubscriptions",newsDataList);
        request.setAttribute("message1",
        "<h3>ConfigurationException Detected</h3>");
        request.setAttribute("message2",
        "<h4>" + e.getMessage() + "</h4>");

        RequestDispatcher dispatcher =
        request.getRequestDispatcher("/NewsSubscriptionForm.jsp");
        dispatcher.forward(request,response);
}
```

7. Return.

```
                return;
```

8. Terminate class.

```
                }
```

Everyplace Intelligent Notification Services - *NewsSubscriptionServlet.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;
import NewsSubscriptionData;
import NewsSubscriptions;
import com.ibm.pvc.we.ins.util.SubscriptionUtility;

/* This is a subscriber servlet which allows a WES user to subscribe
to news content.  This servlet uses the INS API to
subscribe to a publish/subscribe system */

public class NewsSubscriptionServlet extends HttpServlet
{
  private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
  static final String topicNews = "news";
  private NewsSubscriptions newsDataList;
  private int identifier;
  private String userid = null;

  /*
   * Method Name: init
   */
   public void init() throws ServletException
   {
      newsDataList = new NewsSubscriptions();
      identifier = 1;
   }

  /*
   * Method Name: doPost
   */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
   {
      userid = SubscriptionUtility.getUser(request);

      if (request.getParameter("add") != null)
          //Add button was pressed
        newsAddData(request, response);
      else
        if (request.getParameter("refresh") != null)
```

```
                //Refresh button was pressed
          newsDeleteData(request, response);
        else
                //Submit button was pressed
          newsSubscription(request, response);
   }
   //end of doPost method

  /*
   * Method Name: newsAddData
   */
   public void newsAddData(HttpServletRequest request,
   HttpServletResponse response)
   throws ServletException, IOException
   {
      NewsSubscriptionData newsData = new NewsSubscriptionData();
      Integer ident = new Integer(identifier);
      newsData.setId(ident.toString());
      newsData.setTopic(request.getParameter("newstopic"));
      newsData.setSubject(request.getParameter("newssubject"));
      newsData.setTriggerOption(request.getParameter("trigopt"));
      newsData.setContentOption(request.getParameter("contopt"));
      newsDataList.setSubscription(userid,newsData);

      identifier++;
      request.setAttribute("newsSubscriptions",newsDataList);

      RequestDispatcher dispatcher =
          request.getRequestDispatcher("/NewsSubscriptionForm.jsp");
      dispatcher.forward(request, response);
   }

  /*
   * Method Name: newsDeleteData
   */
   public void newsDeleteData(HttpServletRequest request,
   HttpServletResponse response)
   throws ServletException, IOException
   {
      String[] deletes = request.getParameterValues("delete");

      if (deletes != null)
          //Check for Refresh pressed, but no subscriptions checked
      {
        try
        {
          for (int x = 0; x < deletes.length; x++)
          {
            NewsSubscriptionData nsd = newsDataList.getSubscription(userid ,0);
                        //Throws an exception if empty
            for (int i = 0; ; )
            {
              if (nsd.getId().equals(deletes[x]))
              {
                newsDataList.removeSubscription(userid, i);
                break;
              }
              else
              {
                i++;
                try
                {
```

```
                nsd =  newsDataList.getSubscription(userid,i);
              }
              catch (java.lang.ArrayIndexOutOfBoundsException e) {break;}
            }
          }
        }
      }
      catch (java.lang.ArrayIndexOutOfBoundsException e) {}
    }

    request.setAttribute("newsSubscriptions",newsDataList);

    RequestDispatcher dispatcher =
        request.getRequestDispatcher("/NewsSubscriptionForm.jsp");
    dispatcher.forward(request,response);
 }

/*
 * Method Name: newsSubscription
 */
 public void newsSubscription(HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException
 {
      //Get the server name and port number on which this
      //servlet is running. This will be used
      //in the notification message to the user that has
      //requested that the message content be saved.
      //It will be used by the user to retrieve the
      //message from the content store.
      //This method assumes that all servlets are
      //running on the same server.

    final String server = request.getServerName();
    final int port = request.getServerPort();
    String path = request.getRequestURI();

    //Get the IQueue server host and port from the properties file
    String IQueuehost = SubscriptionUtility.getHost();
    String IQueueport = SubscriptionUtility.getPort();

    //Variable Declarations
    Vector topics = new Vector();
    Vector subjects = new Vector();
    Vector contentOptions = new Vector();
        //Save or Don't Save
    Vector triggerOptions = new Vector();
        //Once or Always
    String sqlsel;
        //SQL selector
    String URL;
        //URL for retention notification message

    //Prepare for output
    response.setContentType("text/html");

      //Construct the first part of the URL for any
      //retention notification messages that will be sent.
      //This URL will be passed to the WeatherHandler.
    URL = "http://" + server + ":" + port +
        path.substring(0,path.lastIndexOf("/")+ 1);
```

```
try
{
  NewsSubscriptionData nsd = newsDataList.getSubscription(userid ,0);
          //Throws an exception if empty
  for (int i = 0; ; )
  {
     topics.add(nsd.getTopic());
     subjects.add(nsd.getSubject());
     triggerOptions.add(nsd.getTriggerOption());
     contentOptions.add(nsd.getContentOption());
     i++;
     try
     {
       nsd =  newsDataList.getSubscription(userid,i);
     }
     catch (java.lang.ArrayIndexOutOfBoundsException e) {break;}
  }
}
catch (java.lang.ArrayIndexOutOfBoundsException e)
{
  //No subscriptions were submitted.  Display an error message to the user.
  request.setAttribute("newsSubscriptions",newsDataList);
  request.setAttribute("message1",
          "<h3>No subscriptions were specified</h3>");

  RequestDispatcher dispatcher =
          request.getRequestDispatcher("/NewsSubscriptionForm.jsp");
  dispatcher.forward(request,response);
  return;
}

// Connect to INS by calling a TriggerManager
try
{
  //Set up to register a TriggerHandler
  TriggerManager manager = new SocketClientStub(IQueuehost,IQueueport);
  ContentSource source =
          SimpleContentSource.newSimpleContentSource(topicNews);

  ContentFilter filter;
  NewsHandler triggerHand;
  TriggerID tid;

  for (int i=0; i<topics.size(); i++)
  {
    //Retrieve the trigger handler
    triggerHand = new NewsHandler(userid,
            (String)triggerOptions.elementAt(i),
                  (String)contentOptions.elementAt(i), URL);
    sqlsel =
          "(datasource = '" + topics.elementAt(i)
          + "') AND (headline LIKE '%" + subjects.elementAt(i)+ "%')";
    filter = SqlSelector.newSqlSelector(sqlsel);
    //Call the trigger manager and add the trigger with parameters
    tid = manager.addTrigger(userid, source, filter, triggerHand);
  }                               // end of loop for trigger conditions

  //If no exception occurred, send back a success message
  newsDataList.removeAllSubscriptions(userid);
          //Successful send, remove all subscriptions from table
  request.setAttribute("newsSubscriptions",newsDataList);
  request.setAttribute("message1",
                "<h3>Your subscription has been accepted</h3>");

        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/NewsSubscriptionForm.jsp");
        dispatcher.forward(request,response);
      }                         // end of try clause
      catch ( IQueueException iqe )
      {
        iqe.printStackTrace(System.out);
        request.setAttribute("newsSubscriptions",newsDataList);
        request.setAttribute("message1",
                "<h3>IQueueException Detected</h3>");
        request.setAttribute("message2","<h4>" +
                iqe.getMessage() + "</h4>");

        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/NewsSubscriptionForm.jsp");
        dispatcher.forward(request,response);
      }
      catch ( ConfigurationException e )
      {
        e.printStackTrace(System.out);
        request.setAttribute("newsSubscriptions",newsDataList);
        request.setAttribute("message1",
                "<h3>ConfigurationException Detected</h3>");
        request.setAttribute("message2",
                "<h4>" + e.getMessage() + "</h4>");

        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/NewsSubscriptionForm.jsp");
        dispatcher.forward(request,response);
      }
      return;
   }
   // end of newsSubscription method
}
// end of NewsSubscriptionServlet class
```

Everyplace Intelligent Notification Services - *WeatherSubscriptionServlet.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;
import WeatherSubscriptionData;
import WeatherSubscriptions;
import com.ibm.pvc.we.ins.util.SubscriptionUtility;

/* This is a subscriber servlet which allows a WES user to subscribe
to weather content.  This servlet uses the INS API to
subscribe to a publish/subscribe system */

public class WeatherSubscriptionServlet extends HttpServlet
{
  private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
  static final String topicWeather = "weather";
  private WeatherSubscriptions weatherDataList;
  private int identifier;
  private String userid = null;

  /*
   * Method Name: init
   */
   public void init() throws ServletException
   {
      weatherDataList = new WeatherSubscriptions();
      identifier = 1;
   }

  /*
   * Method Name: doPost
   */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
   {
      userid = SubscriptionUtility.getUser(request);

      if (request.getParameter("add") != null)
          //Add button was pressed
        weatherAddData(request, response);
      else
        if (request.getParameter("refresh") != null)
```

```
              //Refresh button was pressed
        weatherDeleteData(request, response);
      else
              //Submit button was pressed
        weatherSubscription(request, response);
 }                                    // end of doPost method

/*
 * Method Name: weatherAddData
 */
 public void weatherAddData
 (HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
 {
    WeatherSubscriptionData weatherData = new WeatherSubscriptionData();
    Integer ident = new Integer(identifier);
    weatherData.setId(ident.toString());
    weatherData.setCity(request.getParameter("city"));
    weatherData.setState(request.getParameter("state"));
    if (request.getParameter("current") == null)
      weatherData.setCurrent("false");
    else
      weatherData.setCurrent("true");
    if (request.getParameter("forecast") == null)
      weatherData.setForecast("false");
    else
      weatherData.setForecast("true");
    weatherData.setTriggerOption(request.getParameter("trigopt"));
    weatherData.setContentOption(request.getParameter("contopt"));
    weatherDataList.setSubscription(userid,weatherData);

    identifier++;
    request.setAttribute("weatherSubscriptions",weatherDataList);

    RequestDispatcher dispatcher =
        request.getRequestDispatcher("/WeatherSubscriptionForm.jsp");
    dispatcher.forward(request, response);
 }

/*
 * Method Name: weatherDeleteData
 */
 public void weatherDeleteData
 (HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
 {
    String[] deletes = request.getParameterValues("delete");

    if (deletes != null)
        //Check for Refresh pressed, but no subscriptions checked
    {
      try
      {
        for (int x = 0; x < deletes.length; x++)
        {
          WeatherSubscriptionData wsd =
                      weatherDataList.getSubscription(userid ,0);
                      //Throws an exception if empty
          for (int i = 0; ; )
          {
            if (wsd.getId().equals(deletes[x]))
            {
              weatherDataList.removeSubscription(userid, i);
              break;
            }
            else
            {
              i++;
              try
              {
                wsd =  weatherDataList.getSubscription(userid,i);
              }
              catch (java.lang.ArrayIndexOutOfBoundsException e) {break;}
            }
          }
        }
      }
      catch (java.lang.ArrayIndexOutOfBoundsException e) {}
    }

    request.setAttribute("weatherSubscriptions",weatherDataList);

    RequestDispatcher dispatcher =
        request.getRequestDispatcher("/WeatherSubscriptionForm.jsp");
    dispatcher.forward(request,response);
 }

/*
 * Method Name: weatherSubscription
 */
 public void weatherSubscription
 (HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
 {
      //Get the server name and port number on which this
      //servlet is running.  This will be used
      //in the notification message to the user that has
      //requested that the message content be saved.
      //It will be used by the user to retrieve the message from the content store.
      //This method assumes that all servlets are running on the same server.

    final String server = request.getServerName();
    final int port = request.getServerPort();
    String path = request.getRequestURI();

    //Get the IQueue server host and port from the properties file
    String IQueuehost = SubscriptionUtility.getHost();
    String IQueueport = SubscriptionUtility.getPort();

    //Variable Declarations
    Vector cities = new Vector();
    Vector states = new Vector();
    Vector conditions = new Vector();
        //Current conditions
    Vector forecasts = new Vector();
        //Forecast conditions
    Vector contentOptions = new Vector();
        //Save or Don't Save
    Vector triggerOptions = new Vector();
        //Once or Always
    String sqlsel;
        //SQL selector for trigger condition
    String URL;
        //URL for retention notification message
```

```
//Prepare for output
response.setContentType("text/html");

  //Construct the first part of the URL for any retention
  //notification messages that will be sent.
  //This URL will be passed to the WeatherHandler.
URL = "http://" + server + ":" + port +
    path.substring(0,path.lastIndexOf("/")+ 1);

try
{
  WeatherSubscriptionData wsd = weatherDataList.getSubscription(userid ,0);
          //Throws an exception if empty
  for (int i = 0; ; )
  {
     cities.add(wsd.getCity());
     states.add(wsd.getState());
     conditions.add(Boolean.valueOf(wsd.getCurrent()));
     forecasts.add(Boolean.valueOf(wsd.getForecast()));
     triggerOptions.add(wsd.getTriggerOption());
     contentOptions.add(wsd.getContentOption());
     i++;
     try
     {
       wsd =  weatherDataList.getSubscription(userid,i);
     }
     catch (java.lang.ArrayIndexOutOfBoundsException e) {break;}
  }
}
catch (java.lang.ArrayIndexOutOfBoundsException e)
{
  //No subscriptions were submitted.  Display an error message to the user.
  request.setAttribute("weatherSubscriptions",weatherDataList);
  request.setAttribute("message1",
          "<h3>No subscriptions were specified</h3>");

  RequestDispatcher dispatcher =
          request.getRequestDispatcher("/WeatherSubscriptionForm.jsp");
  dispatcher.forward(request,response);
  return;
}

// Connect to INS by calling a TriggerManager
try
{
  //Set up to register a TriggerHandler
  TriggerManager manager = new SocketClientStub(IQueuehost,IQueueport);
  ContentSource source =
          SimpleContentSource.newSimpleContentSource(topicWeather);

  ContentFilter filter;
  WeatherHandler triggerHand;
  TriggerID tid;
  boolean wantCurrentConditions;
  boolean wantForecasts;

  //Retrieve the trigger handler
  for (int i=0; i<cities.size(); i++)
  {
    triggerHand = new WeatherHandler
            (userid, (String)triggerOptions.elementAt(i),
               (String)contentOptions.elementAt(i),
                                URL);
          sqlsel = "(cityname = '" + cities.elementAt(i) + "')";
          if (states.elementAt(i) != "")
            sqlsel = sqlsel + " AND (citystate = '" + states.elementAt(i) + "')";
          wantCurrentConditions = ((Boolean)conditions.elementAt(i)).booleanValue();
          wantForecasts = ((Boolean)forecasts.elementAt(i)).booleanValue();
          if ((wantCurrentConditions) ^ (wantForecasts))
            if (wantCurrentConditions)
              sqlsel = sqlsel + " AND (forecastday IS NULL)";
            else
              sqlsel = sqlsel + " AND (forecastday IS NOT NULL)";

          filter = SqlSelector.newSqlSelector(sqlsel);
          //Call the trigger manager and add the trigger with parameters
          tid = manager.addTrigger(userid, source, filter, triggerHand);
        }                              // end of loop for trigger conditions

        //If no exception occurred, send back a success message
        weatherDataList.removeAllSubscriptions(userid);
                //Successful send, remove all subscriptions from table
        request.setAttribute("weatherSubscriptions",weatherDataList);
        request.setAttribute("message1",
                "<h3>Your subscription has been accepted</h3>");

        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/WeatherSubscriptionForm.jsp");
        dispatcher.forward(request,response);
      }                         // end of try clause
      catch ( IQueueException iqe )
      {
        iqe.printStackTrace(System.out);
        request.setAttribute("weatherSubscriptions",weatherDataList);
        request.setAttribute("message1",
                "<h3>IQueueException Detected</h3>");
        request.setAttribute("message2","<h4>" +
                iqe.getMessage() + "</h4>");

        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/WeatherSubscriptionForm.jsp");
        dispatcher.forward(request,response);
      }
      catch ( ConfigurationException e )
      {
        e.printStackTrace(System.out);
        request.setAttribute("weatherSubscriptions",weatherDataList);
        request.setAttribute("message1",
                "<h3>ConfigurationException Detected</h3>");
        request.setAttribute("message2","<h4>" +
                e.getMessage() + "</h4>");

        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/WeatherSubscriptionForm.jsp");
        dispatcher.forward(request,response);
      }
      return;
   }
   // end of weatherSubscription method
}
// end of WeatherSubscriptionServlet class
```

Everyplace Intelligent Notification Services - *StockSubscriptionServlet.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;
import StockSubscriptionData;
import StockSubscriptions;
import com.ibm.pvc.we.ins.util.SubscriptionUtility;

/* This is a subscriber servlet which allows a WES user to subscribe
to stock content.  This servlet uses the INS API to
subscribe to a publish/subscribe system */

public class StockSubscriptionServlet extends HttpServlet
{
  private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
  static final String topicStocks = "stocks";
  private StockSubscriptions stockDataList;
  private int identifier;
  private String userid = null;

  /*
   * Method Name: init
   */
   public void init() throws ServletException
   {
      stockDataList = new StockSubscriptions();
      identifier = 1;
   }

  /*
   * Method Name: doPost
   */
   public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
   {
      userid = SubscriptionUtility.getUser(request);

      if (request.getParameter("add") != null)
          //Add button was pressed
        stockAddData(request, response);
      else
        if (request.getParameter("refresh") != null)
```

```
              //Refresh button was pressed
        stockDeleteData(request, response);
      else
              //Submit button was pressed
        stockSubscription(request, response);
 }
 // end of doPost method

/*
 * Method Name: stockAddData
 */
 public void stockAddData(HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException
 {
    StockSubscriptionData stockData = new StockSubscriptionData();
    Integer ident = new Integer(identifier);
    stockData.setId(ident.toString());
    stockData.setSymbol(request.getParameter("stocksymbol"));
    stockData.setOp(request.getParameter("stockop"));
    stockData.setPrice(request.getParameter("stockprice"));
    stockData.setTriggerOption(request.getParameter("trigopt"));
    stockData.setContentOption(request.getParameter("contopt"));
    stockDataList.setSubscription(userid,stockData);

    identifier++;
    request.setAttribute("stockSubscriptions",stockDataList);

    RequestDispatcher dispatcher =
        request.getRequestDispatcher("/StockSubscriptionForm.jsp");
    dispatcher.forward(request, response);
 }

/*
 * Method Name: stockDeleteData
 */
 public void stockDeleteData(HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException
 {
    String[] deletes = request.getParameterValues("delete");

    if (deletes != null)
        //Check for Refresh pressed, but no subscriptions checked
    {
      try
      {
        for (int x = 0; x < deletes.length; x++)
        {
          StockSubscriptionData ssd = stockDataList.getSubscription(userid ,0);
                      //Throws an exception if empty
          for (int i = 0; ; )
          {
            if (ssd.getId().equals(deletes[x]))
            {
              stockDataList.removeSubscription(userid, i);
              break;
            }
            else
            {
              i++;
              try
              {
                ssd =  stockDataList.getSubscription(userid,i);
              }
              catch (java.lang.ArrayIndexOutOfBoundsException e) {break;}
            }
          }
        }
      }
      catch (java.lang.ArrayIndexOutOfBoundsException e) {}
    }

    request.setAttribute("stockSubscriptions",stockDataList);

    RequestDispatcher dispatcher =
        request.getRequestDispatcher("/StockSubscriptionForm.jsp");
    dispatcher.forward(request,response);
 }

/*
 * Method Name: stockSubscription
 */
 public void stockSubscription(HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException
 {
      //Get the server name and port number on which this servlet is
      //running.  This will be used
      //in the notification message to the user that has requested
      //that the message content be saved.
      //It will be used by the user to retrieve the message from
      //the content store.
      //This method assumes that all servlets are running on the same server.

    final String server = request.getServerName();
    final int port = request.getServerPort();
    String path = request.getRequestURI();

    //Get the IQueue server host and port from the properties file
    String IQueuehost = SubscriptionUtility.getHost();
    String IQueueport = SubscriptionUtility.getPort();

    //Variable Declarations
    Vector stocksymbol = new Vector();
    Vector stockop = new Vector();
    Vector stockvalue = new Vector();
    Vector contentOptions = new Vector();
        //Save or Don't Save
    Vector triggerOptions = new Vector();
        //Once or Always
    String sqlsel;
        //SQL selector
    String URL;
        //URL for retention notification message

    //Prepare for output
    response.setContentType("text/html");

      //Construct the first part of the URL for any retention
      //notification messages that will be sent.
      //This URL will be passed to the WeatherHandler.
    URL = "http://" + server + ":" + port +
        path.substring(0,path.lastIndexOf("/")+ 1);
```

```
try
{
  StockSubscriptionData ssd = stockDataList.getSubscription(userid ,0);
          //Throws an exception if empty
  for (int i = 0; ; )
  {
     stocksymbol.add(ssd.getSymbol());
     stockop.add(ssd.getOp());
     stockvalue.add(ssd.getPrice());
     triggerOptions.add(ssd.getTriggerOption());
     contentOptions.add(ssd.getContentOption());
     i++;
     try
     {
       ssd =  stockDataList.getSubscription(userid,i);
     }
     catch (java.lang.ArrayIndexOutOfBoundsException e) {break;}
  }
}
catch (java.lang.ArrayIndexOutOfBoundsException e)
{
  //No subscriptions were submitted.  Display an error message to the user.
  request.setAttribute("stockSubscriptions",stockDataList);
  request.setAttribute("message1",
          "<h3>No subscriptions were specified</h3>");

  RequestDispatcher dispatcher =
          request.getRequestDispatcher("/StockSubscriptionForm.jsp");
  dispatcher.forward(request,response);
  return;
}

// Connect to INS by calling a TriggerManager
try
{
  //Set up to register a TriggerHandler
  TriggerManager manager = new SocketClientStub(IQueuehost,IQueueport);
  ContentSource source =
          SimpleContentSource.newSimpleContentSource(topicStocks);

  ContentFilter filter;
  StockHandler triggerHand;
  TriggerID tid;

  for (int i=0; i<stocksymbol.size(); i++)
  {
    //Retrieve the trigger handler
    triggerHand = new StockHandler(userid,
            (String)triggerOptions.elementAt(i),
                  (String)contentOptions.elementAt(i), URL);
    sqlsel = "(symbol = '" + stocksymbol.elementAt(i) +
            "') AND (last " + stockop.elementAt(i) +
            " " + stockvalue.elementAt(i) + ")";
    filter = SqlSelector.newSqlSelector(sqlsel);
    //Call the trigger manager and add the trigger with parameters
    tid = manager.addTrigger(userid, source, filter, triggerHand);
  }                               // end of loop for trigger conditions

  //If no exception occurred, send back a success message
  stockDataList.removeAllSubscriptions(userid);
          //Successful send, remove all subscriptions from table
```

```
        request.setAttribute("stockSubscriptions",stockDataList);
        request.setAttribute("message1",
                "<h3>Your subscription has been accepted</h3>");
        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/StockSubscriptionForm.jsp");
        dispatcher.forward(request,response);
      }                         // end of try clause
      catch ( IQueueException iqe )
      {
        iqe.printStackTrace(System.out);
        request.setAttribute("stockSubscriptions",stockDataList);
        request.setAttribute("message1",
                "<h3>IQueueException Detected</h3>");
        request.setAttribute("message2","<h4>" +
                iqe.getMessage() + "</h4>");

        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/StockSubscriptionForm.jsp");
        dispatcher.forward(request,response);
      }
      catch ( ConfigurationException e )
      {
        e.printStackTrace(System.out);
        request.setAttribute("stockSubscriptions",stockDataList);
        request.setAttribute("message1",
                "<h3>ConfigurationException Detected</h3>");
        request.setAttribute("message2","<h4>" +
                e.getMessage() + "</h4>");
        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/StockSubscriptionForm.jsp");
        dispatcher.forward(request,response);
      }
      return;
   }
   // end of stockSubscription method
}
// end of StockSubscriptionServlet class
```

Everyplace Intelligent Notification Services - *NewsSubscriptionData.java*

```java
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

public class NewsSubscriptionData
{
  private String topic;
  private String subject;
  private String triggerOption;
  private String contentOption;
  private String id;

  /*
   * Method Name: getId
   */
  public String getId()
  {
    return id;
  }

  /*
   * Method Name: setId
   */
  public void setId(String id)
  {
    this.id = id;
  }


  /*
   * Method Name: getTopic
   */
  public String getTopic()
  {
    return topic;
  }

  /*
   * Method Name: setTopic
   */
  public void setTopic(String topic)
  {
    this.topic = topic;
  }
  /*
   * Method Name: getSubject
   */
  public String getSubject()
  {
    return subject;
  }

  /*
   * Method Name: setSubject
   */
  public void setSubject(String subject)
  {
    this.subject = subject;
  }

  /*
   * Method Name: getTriggerOption
   */
  public String getTriggerOption()
  {
    return triggerOption;
  }

  /*
   * Method Name: setTriggerOption
   */
  public void setTriggerOption(String triggerOption)
  {
    this.triggerOption = triggerOption;
  }

  /*
   * Method Name: getContentOption
   */
  public String getContentOption()
  {
    return contentOption;
  }

  /*
   * Method Name: setContentOption
   */
  public void setContentOption(String contentOption)
  {
    this.contentOption = contentOption;
  }
}
```

Everyplace Intelligent Notification Services - *WeatherSubscriptionData.java*

```
/*******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                 *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                             *
 *  Licensed Materials - Property of IBM                                       *
 *                                                                             *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 *******************************************************************************
*/

public class WeatherSubscriptionData
{
  private String city;
  private String state;
  private String current;
  private String forecast;
  private String triggerOption;
  private String contentOption;
  private String id;

  /*
   * Method Name: getId
   */
  public String getId()
  {
    return id;
  }

  /*
   * Method Name: setId
   */
  public void setId(String id)
  {
    this.id = id;
  }


  /*
   * Method Name: getCity
   */
  public String getCity()
  {
    return city;
  }

  /*
   * Method Name: setCity
   */
  public void setCity(String city)
  {
```

```
  this.city = city;
}

/*
 * Method Name: getState
 */
public String getState()
{
  return state;
}

/*
 * Method Name: setState
 */
public void setState(String state)
{
  this.state = state;
}

/*
 * Method Name: getCurrent
 */
public String getCurrent()
{
  return current;
}

/*
 * Method Name: setCurrent
 */
public void setCurrent(String current)
{
  this.current = current;
}

/*
 * Method Name: getForecast
 */
public String getForecast()
{
  return forecast;
}

/*
 * Method Name: setForecast
 */
public void setForecast(String forecast)
{
  this.forecast = forecast;
}

/*
 * Method Name: getTriggerOption
 */
public String getTriggerOption()
{
  return triggerOption;
}
```

```
  /*
   * Method Name: setTriggerOption
   */
  public void setTriggerOption(String triggerOption)
  {
    this.triggerOption = triggerOption;
  }

  /*
   * Method Name: getContentOption
   */
  public String getContentOption()
  {
    return contentOption;
  }

  /*
   * Method Name: setContentOption
   */
  public void setContentOption(String contentOption)
  {
    this.contentOption = contentOption;
  }
}
```

Everyplace Intelligent Notification Services - *StockSubscriptionData.java*

```
/*****************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)              *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved          *
 *                                                                           *
 *  Licensed Materials - Property of IBM                                    *
 *                                                                           *
 *  US Government Users Restricted Rights - Use, duplication or disclosure  *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.           *
 *****************************************************************************
*/

public class StockSubscriptionData
{
  private String symbol;
  private String op;
  private String price;
  private String triggerOption;
  private String contentOption;
  private String id;

  /*
   * Method Name: getId
   */
  public String getId()
  {
    return id;
  }

  /*
   * Method Name: setId
   */
  public void setId(String id)
  {
    this.id = id;
  }


  /*
   * Method Name: getSymbol
   */
  public String getSymbol()
  {
    return symbol;
  }

  /*
   * Method Name: setSymbol
   */
  public void setSymbol(String symbol)
  {
    this.symbol = symbol;
}

/*
 * Method Name: getOp
 */
public String getOp()
{
  return op;
}

/*
 * Method Name: setOp
 */
public void setOp(String op)
{
  this.op = op;
}

/*
 * Method Name: getPrice
 */
public String getPrice()
{
  return price;
}

/*
 * Method Name: setPrice
 */
public void setPrice(String price)
{
  this.price = price;
}

/*
 * Method Name: getTriggerOption
 */
public String getTriggerOption()
{
  return triggerOption;
}

/*
 * Method Name: setTriggerOption
 */
public void setTriggerOption(String triggerOption)
{
  this.triggerOption = triggerOption;
}

/*
 * Method Name: getContentOption
 */
public String getContentOption()
{
  return contentOption;
}
```

```
  /*
   * Method Name: setContentOption
   */
  public void setContentOption(String contentOption)
  {
    this.contentOption = contentOption;
  }
}
```

Everyplace Intelligent Notification Services - *NewsSubscriptions.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

import java.util.*;

public class NewsSubscriptions
{
  private Vector newsSubscriptionList;
  private Hashtable subHT;
  private NewsSubscriptionData subscription;

  /*
   * Constructer: NewsSubscriptions
   */
   public NewsSubscriptions()
   {
     subHT = new Hashtable();
   }

  /*
   * Method Name: getSubscription
   */
  public NewsSubscriptionData getSubscription(String userid, int i)
  throws ArrayIndexOutOfBoundsException
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       newsSubscriptionList = (Vector)subHT.get(userid);
       return (NewsSubscriptionData)newsSubscriptionList.elementAt(i);
    }
    else
      throw new ArrayIndexOutOfBoundsException();
  }

  /*
   * Method Name: setSubscription
   */
  public void setSubscription(String userid, NewsSubscriptionData subscription)
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
      newsSubscriptionList = (Vector)subHT.get(userid);
      newsSubscriptionList.add(subscription);
    }
    else
    {
      newsSubscriptionList = new Vector();
      newsSubscriptionList.add(subscription);
      subHT.put(userid,newsSubscriptionList);
    }
  }

  /*
   * Method Name: removeSubscription
   */
  public void removeSubscription(String userid, int i)
  throws ArrayIndexOutOfBoundsException
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       newsSubscriptionList = (Vector)subHT.get(userid);
       newsSubscriptionList.remove(i);
    }
    else
      throw new ArrayIndexOutOfBoundsException();
  }

  /*
   * Method Name: removeAllSubscriptions
   */
  public void removeAllSubscriptions(String userid)
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       newsSubscriptionList = (Vector)subHT.get(userid);
       newsSubscriptionList = null;
           //Allow the subscription List to be garbage collected
       subHT.remove(userid);
           //Remove the userid from the Subscriptions Hashtable
    }
  }
}
```

Everyplace Intelligent Notification Services - *WeatherSubscriptions.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                 *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved             *
 *                                                                             *
 *  Licensed Materials - Property of IBM                                       *
 *                                                                             *
 *  US Government Users Restricted Rights - Use, duplication or disclosure     *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.              *
 ******************************************************************************
*/

import java.util.*;

public class WeatherSubscriptions
{
  private Vector weatherSubscriptionList;
  private Hashtable subHT;
  private WeatherSubscriptionData subscription;

  /*
   * Constructer: WeatherSubscriptions
   */
   public WeatherSubscriptions()
   {
     subHT = new Hashtable();
   }

  /*
   * Method Name: getSubscription
   */
  public WeatherSubscriptionData getSubscription(String userid, int i)
  throws ArrayIndexOutOfBoundsException
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       weatherSubscriptionList = (Vector)subHT.get(userid);
       return (WeatherSubscriptionData)weatherSubscriptionList.elementAt(i);
    }
    else
      throw new ArrayIndexOutOfBoundsException();
  }

  /*
   * Method Name: setSubscription
   */
  public void setSubscription(String userid, WeatherSubscriptionData subscription)
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
      weatherSubscriptionList = (Vector)subHT.get(userid);
      weatherSubscriptionList.add(subscription);
    }
    else
    {
      weatherSubscriptionList = new Vector();
      weatherSubscriptionList.add(subscription);
      subHT.put(userid,weatherSubscriptionList);
    }
  }

  /*
   * Method Name: removeSubscription
   */
  public void removeSubscription(String userid, int i)
  throws ArrayIndexOutOfBoundsException
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       weatherSubscriptionList = (Vector)subHT.get(userid);
       weatherSubscriptionList.remove(i);
    }
    else
      throw new ArrayIndexOutOfBoundsException();
  }

  /*
   * Method Name: removeAllSubscriptions
   */
  public void removeAllSubscriptions(String userid)
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       weatherSubscriptionList = (Vector)subHT.get(userid);
       weatherSubscriptionList = null;
           //Allow the subscription List to be garbage collected
       subHT.remove(userid);
           //Remove the userid from the Subscriptions Hashtable
    }
  }
}
```

Everyplace Intelligent Notification Services - *StockSubscriptions.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

import java.util.*;

public class StockSubscriptions
{
  private Vector stockSubscriptionList;
  private Hashtable subHT;
  private StockSubscriptionData subscription;

  /*
   * Constructer: StockSubscriptions
   */
   public StockSubscriptions()
   {
     subHT = new Hashtable();
   }

  /*
   * Method Name: getSubscription
   */
  public StockSubscriptionData getSubscription(String userid, int i)
  throws ArrayIndexOutOfBoundsException
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       stockSubscriptionList = (Vector)subHT.get(userid);
       return (StockSubscriptionData)stockSubscriptionList.elementAt(i);
    }
    else
      throw new ArrayIndexOutOfBoundsException();
  }

  /*
   * Method Name: setSubscription
   */
  public void setSubscription(String userid, StockSubscriptionData subscription)
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
      stockSubscriptionList = (Vector)subHT.get(userid);
      stockSubscriptionList.add(subscription);
    }
    else
    {
      stockSubscriptionList = new Vector();
      stockSubscriptionList.add(subscription);
      subHT.put(userid,stockSubscriptionList);
    }
  }

  /*
   * Method Name: removeSubscription
   */
  public void removeSubscription(String userid, int i)
  throws ArrayIndexOutOfBoundsException
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       stockSubscriptionList = (Vector)subHT.get(userid);
       stockSubscriptionList.remove(i);
    }
    else
      throw new ArrayIndexOutOfBoundsException();
  }

  /*
   * Method Name: removeAllSubscriptions
   */
  public void removeAllSubscriptions(String userid)
  {
    if ((!subHT.isEmpty()) && (subHT.containsKey(userid)))
    {
       stockSubscriptionList = (Vector)subHT.get(userid);
       stockSubscriptionList = null;
           //Allow the subscription List to be garbage collected
       subHT.remove(userid);
           //Remove the userid from the Subscriptions Hashtable
    }
  }
}
```

Everyplace Intelligent Notification Services - trigger handler sample code walkthrough

1. The Trigger Handler is issued to process the trigger for output to the subscriber when and if a trigger match is detected by the iQueueServer. When the Trigger Handler is instantiated by the subscription servlet, it is provided with information about the options specified by the user for content and trigger persistence. When a match is made, the handleMatch method of the trigger handler is invoked by the iQueue server. The trigger handler uses content information and user options to format a message and transmit it to the Universal Notification Dispatcher.

2. Import the necessary native methods and the Everyplace Intelligent Notification Services package.
   - `import java.io.*;`
   - `import java.util.*;`
   - `import com.ibm.pvc.we.ins.*;`
   - `com.ibm.pvc.we.ins.util.SubscriptionUtility`

3. Create the public class extending TriggerHandler

```
public class DataHandler extends TriggerHandler
{
```

4. Create private Strings to represent userid, trigopt, contopt, and content retention URL.

```
  private String toUserid;
  private String triggerOption;
  private String contentOption;
  private String notificationURL;
```

5. Construct the public class using userid, trigopt, contopt, and content retention URL.

```
  public DataHandler(String userid, String trigopt, String contopt, String url)
  {
```

6. Set the values of the private Strings

```
    This.toUserid = userid;
    This.triggerOption = trigopt;
    This.contentOption = contopt;
        This.notificationURL = url;
  }
```

7. Override the handleMatch method of the TriggerHandler class.

```
    public TriggerResult handleMatch(ContentBody cb)
    {
```

8. Instantiate the Strings, StringBuffers and integers needed to transfer data.

```
      String contentString;
      String notificationString;
      String secondNotificationString;
      String value;
      StringBuffer content = new StringBuffer();
      StringBuffer message = new StringBuffer();
      StringBuffer message2 = new StringBuffer();
      Int tResult;
```

9. Set the TriggerResult trigger option.

```
    if (triggerOption.equalsIgnoreCase("once"))
    tResult = TriggerResult.STOP;
else if (triggerOption.equalsIgnoreCase("always"))
     tResult = TriggerResult.CONTINUE;
   else
     tResult = TriggerResult.SAVE_AND_CONTINUE
```

10. Check to see if Content Option equals "Save."

```
if (triggerOption.equalsIgnoreCase("save"))
{
```

11. If the Content Option equals "Save", attempt to write the content as XML to "content" String.
    1. Write data to "content" String.

```
 try
 {
   content.append
           (<?xml version=\"1.0\" encoding=\"UTF-8\"?\n");
   content.append("<data>");
   content.append("<name>" +
           cb.getString("name") + "</name>\n");
   content.append("<value>" +
           cb.getString("value") + "<value>\n");
   content.append("</data>\n");
}
```

    2. Catch IqueueException.

```
catch (IQueueException iqe)
{
  iqe.printStackTrace(System.out);
  contentString = iqe.toString();
}
```

    3. Create message to be sent by UND.

```
message2 = new StringBuffer();
message.append
    ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
message.append("<message>\n");
message.append("<to>" + toUserid +
    "</to>\n");
message.append("<from>" + toUserid +
    "</from>\n");
message.append
    ("<subject>Newslt;/subject>\n");
message.append("<text> Follow this URL " +
    notificationURL + "ContTransServlet?pageid=");
if (SubscriptionUtility.isSecure())
    //Are authorization services enabled?
  message2.append(" for details </text>\n");
          //Yes, do not need to pass userid on request
else
  message2.append("&userid=" + toUserid +
          " for details </text>\n"); //No, include userid on request
message2.append("</message>");
notificationString = message.toString();
secondNotificationString = message2.toString();
```

4. Send Trigger Result object to UND.

```
      return TriggerResult.newRetentionSpecification
          (contentString, notificationString, secondNotificationString,
       "NOT_SAVED", 15, tResult);
  }
```

12. If Content Option does not equal "Save,"
    1. Create message to be sent by UND.

```
      else
      {
        try
        {
         message.append
                 ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
         message.append("<message>\n");
         message.append("<to>" + toUserid +
                 "</to>\n");
         message.append("<from>" + toUserid +
                 "</from>\n");
         message.append
                 ("<subject>STOCK</subject>\n");
         message.append("<text>" +
                 cb.getString("symbol") + " = " + value + "</text>\n");
         message.append("</message>");
         notificationString = message.toString();
       }
```

    2. Catch IqueueException.

```
        catch (IQueueException iqe)
        {
          iqe.printStackTrace(System.out);
          notificationString = iqe.toString();
        }
```

    3. Send Trigger Result object to UND.

```
    return TriggerResult.newNotificationSpecification
        (notificationString, tResult);
    }
  }
}
```

Everyplace Intelligent Notification Services - *NewsHandler.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

import java.io.*;
import java.util.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;
import com.ibm.pvc.we.ins.util.SubscriptionUtility;

/* This is a trigger handler for a news subscription */

public class NewsHandler extends TriggerHandler
{
   private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
   private String triggerOption;
   private String contentOption;
   private String toUserid;
   private String notificationURL;

   public NewsHandler(String userid, String trigopt, String contopt, String url)
   {
      this.triggerOption = trigopt;
      this.contentOption = contopt;
      this.toUserid = userid;
      this.notificationURL = url;

   }

   public TriggerResult handleMatch(ContentBody cb)
   {
      String contentString;
      String notificationString;
      String secondNotificationString;
      String value;
      StringBuffer message = new StringBuffer();
      StringBuffer message2;
      int tResult;

      //Set the Trigger Option based on user input
      if (triggerOption.equals("once"))
        tResult = TriggerResult.STOP;
      else
          if (triggerOption.equals("always"))
        tResult = TriggerResult.CONTINUE;
      else
        tResult = TriggerResult.SAVE_AND_CONTINUE;

    if (contentOption.equals("save"))
     {
        StringBuffer content = new StringBuffer();
        try
        {
           content.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
           content.append("<news>\n");
           content.append("<body>\n");
           content.append("<body.head>\n");
           content.append("<headline>\n");
           content.append("<hl1>" + cb.getString("headline") +
                       "</hl1>\n");
           content.append("</headline>\n");
           content.append("<dateline>\n");
           content.append("<story.date>" + cb.getString("dateline") +
                       "</story.date>\n");
           content.append("</dateline>\n");
           content.append("</body.head>\n");
           content.append("<body.content>" + cb.getString("body.content") +
                       "</body.content>\n");
           content.append("</body>");
           content.append("</news>");
           contentString = content.toString();
        }
        catch (IQueueException iqe)
        {
           iqe.printStackTrace(System.out);
           contentString = iqe.toString();
        }
        message2 = new StringBuffer();
        message.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
        message.append("<message>\n");
        message.append("<to>" + toUserid + "</to>\n");
        message.append("<from>" + toUserid + "</from>\n");
        message.append("<subject>News</subject>\n");
        message.append("<text> Follow this URL " + notificationURL +
                "ContTransServlet?pageid=");
        if (SubscriptionUtility.isSecure())
                //Are authorization services enabled?
          message2.append(" for details </text>\n");
              //Yes, do not need to pass userid on request
        else
          message2.append("&userid=" + toUserid +
                      " for details </text>\n");
              //No, include userid on request
        message2.append("</message>");
        notificationString = message.toString();
        secondNotificationString = message2.toString();

        return TriggerResult.newRetentionSpecification(contentString,
                notificationString, secondNotificationString, "NOT_SAVED",
                15, tResult);
     }
     else
         // Option was chosen to not save the message
```

```
       {
          try
          {
            message.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
            message.append("<message>\n");
            message.append("<to>" + toUserid + "</to>\n");
            message.append("<from>" + toUserid + "</from>\n");
            message.append("<subject>News</subject>\n");
            message.append("<text>" + cb.getString("headline") + "\n" +
                        cb.getString("dateline") + "</text>\n");
            message.append("</message>");
            notificationString = message.toString();
          }
          catch (IQueueException iqe)
          {
            iqe.printStackTrace(System.out);
            notificationString = iqe.toString();
          }
          return
                  TriggerResult.newNotificationSpecification
                  (notificationString,tResult);
       }
   }
   //End of handleMatch method
}
```

Everyplace Intelligent Notification Services - *WeatherHandler.java*

```
/*******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                 *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved             *
 *                                                                             *
 *  Licensed Materials - Property of IBM                                       *
 *                                                                             *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.              *
 *******************************************************************************
*/

import java.io.*;
import java.util.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;
import com.ibm.pvc.we.ins.util.SubscriptionUtility;

/* This is a trigger handler for a weather subscription */

public class WeatherHandler extends TriggerHandler
{
   private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
   private String triggerOption;
   private String contentOption;
   private String toUserid;
   private String datasource;
   private String notificationURL;

   public WeatherHandler(String userid, String trigopt, String contopt, String url)
   {
      this.triggerOption = trigopt;
      this.contentOption = contopt;
      this.toUserid = userid;
      this.notificationURL = url;
   }

   public TriggerResult handleMatch(ContentBody cb)
   {
      String contentString;
      String notificationString;
      String secondNotificationString;
      String value;
      StringBuffer message = new StringBuffer();
      StringBuffer message2;
      int tResult;

      //Set the Trigger Option based on user input
      if (triggerOption.equals("once"))
        tResult = TriggerResult.STOP;
      else
          if (triggerOption.equals("always"))
          tResult = TriggerResult.CONTINUE;
        else
        tResult = TriggerResult.SAVE_AND_CONTINUE;

    if (contentOption.equals("save"))
     {
        StringBuffer content = new StringBuffer();
        try
        {
           content.append
                       ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
           content.append("<WEATHER>\n");
           if (cb.itemExists("timestamp"))
             content.append("<TIMESTAMP>" +
                         cb.getString("timestamp") + "</TIMESTAMP>\n");
           content.append("<CITY NAME = '"+
                       cb.getString("cityname") + "'");
           content.append(" STATE = '" + cb.getString("citystate") +
                       "'" + ">\n");
           if (cb.itemExists("forecastday"))
           {
              content.append("<FORECAST DAY = " +
                              "'" + cb.getString("forecastday") + "'");
              content.append(" NAME = '" +
                              cb.getString("forecastname") + "'>\n");
              if (cb.itemExists("condition"))
                content.append("<CONDITION>" +
                                cb.getString("condition") + "</CONDITION>\n");
              if (cb.itemExists("high"))
                content.append("<HIGH F = '" +
                                cb.getString("high") + "'></HIGH>\n");
              if (cb.itemExists("low"))
                content.append("<LOW F = '" +
                                cb.getString("low") + "'></LOW>\n");
              if (cb.itemExists("sunrise"))
                content.append("<SUNRISE>" +
                                cb.getString("sunrise") + "</SUNRISE>\n");
              if (cb.itemExists("sunset"))
                content.append("<SUNSET>" +
                                cb.getString("sunset") + "</SUNSET>\n");
              content.append("</FORECAST>\n");
           }
           else
           {
             if (cb.itemExists("temperature"))
               content.append("<TEMPERATURE F = '" +
                               cb.getString("temperature")+
                               "'></TEMPERATURE>\n");
             if (cb.itemExists("wind"))
               content.append("<WIND>" +
                               cb.getString("wind") + "</WIND>\n");
             if (cb.itemExists("humidity"))
               content.append("<HUMIDITY>" +
                               cb.getString("humidity") + "</HUMIDITY>\n");
             if (cb.itemExists("pressure"))
               content.append("<BAROMETER>" +
                               cb.getString("pressure") + "</BAROMETER>\n");
             content.append("<CONDITION>" +
                         cb.getString("condition") + "</CONDITION>\n");
             if (cb.itemExists("sunrise"))
               content.append("<SUNRISE>" +
                               cb.getString("sunrise") + "</SUNRISE>\n");
             if (cb.itemExists("sunset"))
```

```
              content.append("<SUNSET>" +
                              cb.getString("sunset") + "</SUNSET>\n");
          }
          content.append("</CITY>\n");
          content.append("</WEATHER>\n");
          contentString = content.toString();
       }
       catch (IQueueException iqe)
       {
          iqe.printStackTrace(System.out);
          contentString = iqe.toString();
       }
       message2 = new StringBuffer();
       message.append
               ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
       message.append("<message>\n");
       message.append("<to>" + toUserid +
               "</to>\n");
       message.append("<from>" + toUserid +
               "</from>\n");
       message.append
               ("<subject>WEATHER</subject>\n");
       message.append("<text> Follow this URL " +
               notificationURL + "ContTransServlet?pageid=");
       if (SubscriptionUtility.isSecure())
               //Are authorization services enabled?
         message2.append(" for details </text>\n");
                     //Yes, do not need to pass userid on request
       else
         message2.append("&userid=" + toUserid +
                     " for details </text>\n");
                     //No, include userid on request
       message2.append("</message>");
       notificationString = message.toString();
       secondNotificationString = message2.toString();

       return TriggerResult.newRetentionSpecification
               (contentString, notificationString, secondNotificationString,
               "NOT_SAVED", 15, tResult);
    }
    else
        // Option was chosen to not save the message
    {
       try
       {
         message.append
                     ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
         message.append("<message>\n");
         message.append("<to>" + toUserid +
                     "</to>\n");
         message.append("<from>" + toUserid +
                     "</from>\n");
         message.append
                     ("<subject>WEATHER</subject>\n");
         message.append("<text>" + cb.getString("cityname") +
                     " " + cb.getString("citystate") + " ");
         if (cb.itemExists("timestamp"))
           message.append(cb.getString("timestamp") + "\n");
          if (cb.itemExists("forecastday"))
          {
             message.append("Forecast for " +
                                cb.getString("forecastname") + " ");
                message.append(cb.getString("forecastday") + "\n");
                if (cb.itemExists("condition"))
                  message.append("Forecast Condition: " +
                                  cb.getString("condition") + "\n");
                if (cb.itemExists("high"))
                  message.append("Forecast high: " +
                                  cb.getString("high") + "\n");
                if (cb.itemExists("low"))
                  message.append("Forecast low: " +
                                  cb.getString("low") + "\n");
                if (cb.itemExists("sunrise"))
                  message.append("Sunrise: " +
                                  cb.getString("sunrise") + "\n");
                if (cb.itemExists("sunset"))
                  message.append("Sunset: " +
                                  cb.getString("sunset") + "\n");
             }
             else
             {
               if (cb.itemExists("temperature"))
                 message.append("Temperature: " +
                                 cb.getString("temperature") + "\n");
               if (cb.itemExists("wind"))
                 message.append("Wind: " +
                                 cb.getString("wind") + "\n");
               if (cb.itemExists("humidity"))
                 message.append("Humidity: " +
                                 cb.getString("humidity") + "\n");
               if (cb.itemExists("pressure"))
                 message.append("Pressure: " +
                                 cb.getString("pressure") + "\n");
               message.append("Condition: " +
                           cb.getString("condition") + "\n");
               if (cb.itemExists("sunrise"))
                 message.append("Sunrise: " +
                                 cb.getString("sunrise") + "\n");
               if (cb.itemExists("sunset"))
                 message.append("Sunset: " +
                                 cb.getString("sunset") + "\n");
             }
            message.append("</text>\n");
            message.append("</message>");
            notificationString = message.toString();
          }
          catch (IQueueException iqe)
          {
            iqe.printStackTrace(System.out);
            notificationString = iqe.toString();
          }
          return
                  TriggerResult.newNotificationSpecification
                  (notificationString,tResult);
       }
   }
   //End of handleMatch method
}
//End of WeatherHandler class
```

Everyplace Intelligent Notification Services - *StockHandler.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)               *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

import java.io.*;
import java.util.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;
import com.ibm.pvc.we.ins.util.SubscriptionUtility;

/* This is a trigger handler for a stock subscription */

public class StockHandler extends TriggerHandler
{
   private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
   private String triggerOption;
   private String contentOption;
   private String toUserid;
   private String stocktype = "last";
   //Default to last
   private String notificationURL;


   public StockHandler(String userid, String trigopt,
   String contopt, String url)
   {
      this.triggerOption = trigopt;
      this.contentOption = contopt;
      this.toUserid = userid;
      this.notificationURL = url;
   }

   public TriggerResult handleMatch(ContentBody cb)
   {
      String contentString;
      String notificationString;
      String secondNotificationString;
      String value,change,percentage;
      StringBuffer message = new StringBuffer();
      StringBuffer message2;
      int tResult;

      //Set the Trigger Option based on user input
      if (triggerOption.equals("once"))
```

```
  tResult = TriggerResult.STOP;
else
    if (triggerOption.equals("always"))
    tResult = TriggerResult.CONTINUE;
  else
    tResult = TriggerResult.SAVE_AND_CONTINUE;

if (contentOption.equals("save"))
 {
    StringBuffer content = new StringBuffer();
    try
    {
       value = String.valueOf(cb.getFloat(stocktype));
       change = String.valueOf(cb.getFloat("change"));
       percentage = String.valueOf(cb.getFloat
                   ("percentage_change"));
       content.append
                   ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
       content.append("<quotes>\n");
       content.append("<stockquote>\n");
       content.append("<companyName>" + cb.getString("symbol") +
                   "</companyName>\n");
       content.append("<symbol>" + cb.getString("symbol") +
                   "</symbol>\n");
       content.append("<curValueUSD>" + value +
                   "</curValueUSD>\n");
       content.append("<curChangeUSD>" + change +
                   "</curChangeUSD>\n");
       content.append("<curChangePct>" + percentage +
                   "</curChangePct>\n");
       content.append("<timestamp>" + cb.getString("timestamp") +
                   "</timestamp>\n");
       content.append("</stockquote>\n");
       content.append("</quotes>");
       contentString = content.toString();
    }
    catch (IQueueException iqe)
    {
       iqe.printStackTrace(System.out);
       contentString = iqe.toString();
    }
    message2 = new StringBuffer();
    message.append
            ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
    message.append("<message>\n");
    message.append("<to>" + toUserid +
            "</to>\n");
    message.append("<from>" + toUserid +
            "</from>\n");
    message.append
            ("<subject>STOCK</subject>\n");
    message.append("<text> Follow this URL " +
            notificationURL + "ContTransServlet?pageid=");
    if (SubscriptionUtility.isSecure())
            //Are authorization services enabled?
      message2.append(" for details </text>\n");
                  //Yes, do not need to pass userid on request
    else
      message2.append("&userid=" + toUserid +
                         " for details </text>\n");
                         //No, include userid on request
          message2.append("</message>");
          notificationString = message.toString();
          secondNotificationString = message2.toString();

          return TriggerResult.newRetentionSpecification
                  (contentString, notificationString, secondNotificationString,
                  "NOT_SAVED", 15, tResult);
       }
       else
           // Option was chosen to not save the message
       {
          try
          {
            value = String.valueOf(cb.getFloat(stocktype));
            message.append
                         ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
            message.append("<message>\n");
            message.append("<to>" + toUserid +
                         "</to>\n");
            message.append("<from>" + toUserid +
                         "</from>\n");
            message.append
                         ("<subject>STOCK</subject>\n");
            message.append("<text>" + cb.getString("symbol") +
                         " = " + value + "</text>\n");
            message.append("</message>");
            notificationString = message.toString();
          }
          catch (IQueueException iqe)
          {
            iqe.printStackTrace(System.out);
            notificationString = iqe.toString();
          }
          return
                  TriggerResult.newNotificationSpecification
                  (notificationString,tResult);
       }
   }
   //End of handleMatch method
}
//End of StockHandler class
```

Everyplace Intelligent Notification Services - *ContTransServlet.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
 */

import java.io.*;
import java.util.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import org.w3c.dom.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.pvc.we.ins.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.util.SubscriptionUtility;

/* This is a content retrieval servlet which allows a WES user to
retrieve content which has been saved on a match to a subscription.
This servlet uses the INS API to retrieve the content from the
persistent content store. */

public class ContTransServlet extends HttpServlet
{
  private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
  {
    String userID = null;
    String string_pageID = null;
    long pageID = 0;
    String content = null;
    String stylesheet = null;

    //Get the IQueue server host and port from the properties file
    String IQueuehost = SubscriptionUtility.getHost();
    String IQueueport = SubscriptionUtility.getPort();

    //Get the name of the stylesheet from the properties file
    stylesheet = SubscriptionUtility.getStyleSheet();

    // Prepare for output
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    // retrieve the content's pageid from the url query string
    string_pageID = request.getParameter("pageid");
```

```
userID = SubscriptionUtility.getUser(request);
    //Retrieve the userid

try
{
  pageID = Long.parseLong(string_pageID);
}
catch (NumberFormatException nfe)
{
   //Invalid pageID returned.  An error occurred storing content.
       //Display an error message to the user.
   out.println("<html>" +
       "<head><title> Content Retrieval Receipt </title></head>"
       + "<body  bgcolor=\"#FFFFFF\">");
   out.println
       ("<h3>Error occurred storing the content."
       + "The content was not saved.</h3>"
       + "</body></html>");
   out.close();
   return;
}

// call the trigger manager and fetch the content from the database
try
{
  TriggerManager tm = new SocketClientStub(IQueuehost, IQueueport);
  try
  {
    content = tm.fetchPersistentContent(pageID, userID);
    if (( content == null ) || ( content.trim().length() == 0 ))
    {
      //No content stored at specified pageid.
              //Display an error message to the user.
      out.println("<html>" +
       "<head><title> Content Retrieval Receipt </title>"
                    + "</head>" + "<body  bgcolor=\"#FFFFFF\">");
      out.println("<h3>No content stored at pageID " + pageID
              + "</h3>" + "</body></html>");
      out.close();
      return;
    }
    // store content into a file: content[pageID].xml
    FileWriter write_content = new FileWriter("content" + pageID + ".xml");
    write_content.write(content);
    write_content.close();

    // Use the static TransformerFactory.newInstance() method to instantiate
    // a TransformerFactory. The javax.xml.transform.TransformerFactory
    // system property setting determines the actual class to instantiate --
    // org.apache.xalan.transformer.TransformerImpl.
    // TransformerFactory tFactory = TransformerFactory.newInstance();
    // Use the TransformerFactory to instantiate a Transformer that will work
    // with the stylesheet you specify. This method call also processes the
    // stylesheet into a compiled Templates object. Then do the transformation.
    try
    {
      Transformer transformer = tFactory.newTransformer
              (new StreamSource(stylesheet));
      transformer.transform(new StreamSource("content" + pageID + ".xml"),
              new StreamResult(out));
    }
    catch (TransformerException tce)
    {
     tce.printStackTrace();
     out.println("<html>" +
           "<head><title> Content Retrieval Receipt </title>"
                     + "</head>" +
           "<body  bgcolor=\"#FFFFFF\">");
     out.println("<h3>TransformerException Detected</h3>");
     out.println("<h4>" + tce.getMessage() +
         "</h4></body></html>");
    }
   }
   catch (IQueueException iqe)
   {
    iqe.printStackTrace();
    out.println("<html>" +
          "<head><title> Content Retrieval Receipt </title>"
                    + "</head>" +
          "<body  bgcolor=\"#FFFFFF\">");
    out.println("<h3>IQueueException Detected</h3>");
    out.println("<h4>" + iqe.getMessage() +
        "</h4></body></html>");
   }
  }
  catch (ConfigurationException ce)
  {
   ce.printStackTrace();
   out.println("<html>" +
         "<head><title> Content Retrieval Receipt </title>"
                 + "</head>" +
         "<body  bgcolor=\"#FFFFFF\">");
   out.println("<h3>ConfigurationException Detected</h3>");
   out.println("<h4>" + ce.getMessage() +
     "</h4></body></html>");
  }
  out.close();
  File contentFile = new File("content" + pageID + ".xml");
  try
  {
   contentFile.delete();
     //Delete the content file that was created.
  }
  catch (SecurityException se)
  {}
    //Do nothing if the file could not be deleted.
 }
}
```

Everyplace Intelligent Notification Services - *XMLContentAdapter.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
import java.io.*;
import java.util.*;

import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.ErrorHandler;
import org.xml.sax.Locator;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;

import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.pvc.we.ins.*;

public class XMLContentAdapter
{
  private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
  private String parserClass = "org.apache.xerces.parsers.SAXParser";
  private ContentBody cb;
  private TriggerManager manager;
  private ContentSource source;
  private String directory;
  /*********************************************************************/
  /* Name of the properties file containing the host
  /* name and port number of the IQueue server */
  /********************************************************************/
  private String IQproperties = "IQ";

  public XMLContentAdapter(String cs)
  {
    //Get the IQueue server host and port from the properties file
    PropertyResourceBundle pr = (PropertyResourceBundle)
        PropertyResourceBundle.getBundle(IQproperties);
    String IQueuehost = pr.getString("iqSocketClientStub.host");
    String IQueueport = pr.getString("iqSocketClientStub.port");
    try
     {
       //Set up to register a TriggerHandler
       manager = new SocketClientStub(IQueuehost,IQueueport);
       //This will change, perhaps to the names of the different
           //sources such as ap.business, etc.
       source = SimpleContentSource.newSimpleContentSource(cs);
           //Register the source of the content
     }
```

```
   catch (ConfigurationException ce)
   {
     System.out.println(ce.getMessage());
     return;
   }

}             //End of constructer

public static void main(String[] args)
{
  //Pass in the name of the directory containing the XML files
  String contentSource = args[0];
  String uri = args[1];
  XMLContentAdapter xmlParser = new XMLContentAdapter(contentSource);
  xmlParser.getXML(uri);
}             //End of main method

public void getXML(String uri)
{
  File dir = new File(uri);
      //Convert the name of the directory into a File descriptor
  File[] fileList = dir.listFiles();
      //Obtain list of files in the directory
  if (fileList == null)
      //No files contained in the directory
  {
      //Display an error message to the user
    System.out.println
        ("Directory specified does not contain any files.\n" +
        "Directory specified: " + uri);
    return;
  }
      //End of no files in the directory

  parseXML(fileList);
      //Parse each of the xml files
}
public void parseXML(File[] fileList)
{
      //Get instances of our handlers for parsing
   ContentHandler contentHandler = new XMLContentHandler();
   ErrorHandler errorHandler = new XMLErrorHandler();
   try
   {
      XMLReader parser = XMLReaderFactory.createXMLReader(parserClass);
              //Instantiate a parser
      parser.setContentHandler(contentHandler);
              //Register the content handler
      parser.setErrorHandler(errorHandler);
              //Register the error handler
      //parser.setFeature("http://xml.org/sax/features/validation", true);
              //Enable validation
      for (int i=0; i<fileList.length; i++)
      {
         if (isXML(fileList[i]))
                 //Is this an XML file
           parser.parse("file:///" + fileList[i].getAbsolutePath());
                       //Parse the document
      }
   }
   catch (IOException ioe)
   {
       System.out.println("Error reading URI: " + ioe.getMessage());
     }
     catch (SAXException se)
     {
          System.out.println("Error in parsing: " + se.getMessage());
     }
  }             //End of parseXML method

  public boolean isXML(File f)
  {
    String filename = f.getName();
    int x = filename.lastIndexOf('.');
    if(x>0 && x<filename.length()-1)
    {
      if (filename.substring(x+1).toLowerCase().equals("xml"))
        return true;
    }
    return false;
  }

/*
 * Module Name: XMLContentHandler.java
 * Copyright  (C): IBM Corp. 2001
 * Author:
 * Modification History:
 *
 */

public class XMLContentHandler implements ContentHandler
{
   private Locator location;
   private String elementName;
   private String elementData;

   public void setDocumentLocator(Locator locator)
   {
      this.location = locator;
   }

   public void startDocument() throws SAXException
   {
      System.out.println("Parsing begins...");
      cb = new ContentBody();
          //Instantiate the content body
   }

   public void endDocument() throws SAXException
   {
      System.out.println("Parsing ends...");
      try
      {
        manager.fireMatchingTriggers(source,cb);
                //Fire matching triggers
      }
      catch (IQueueException iqe)
      {
         System.out.println(iqe.getMessage());
      }
      cb = null;
          //Clean up content body
   }
```

```
public void processingInstruction(String target, String data)
throws SAXException
{
   System.out.println("PI: Target: " + target + " and Data: " + data);
}

public void startPrefixMapping(String prefix, String uri)
{
   System.out.println("Mapping starts for prefix " + prefix +
       " mapped to URI " + uri);
}

public void endPrefixMapping(String prefix)
{
   System.out.println("Mapping ends for prefix " + prefix);
}

public void startElement(String namespaceURI, String localName,
String rawName, Attributes atts) throws SAXException
{
   String attribName;
       //elementName + local attribute name
   System.out.println("startElement: " + localName);
   elementName = localName;
       //Store name of element for possible use by characters method
   elementData = "";
       //Clear out all data for previous element
   if (!namespaceURI.equals(""))
   {
     System.out.println(" in namespace " + namespaceURI +
             "(" + rawName + ")");
   }
   for (int i=0; i<atts.getLength(); i++)
   {
     System.out.println("Attribute: " + atts.getLocalName(i) +
             "=" + atts.getValue(i));
     attribName = elementName + atts.getLocalName(i);
     cb.setString(attribName.trim(),atts.getValue(i).trim());
   }
}

public void endElement(String namespaceURI, String localName,
String rawName) throws SAXException
{
   System.out.println("endElement: " + localName + "\n");
   if (!(elementData.equals("")))
   {
      try
      {
         float fvalue = (Float.valueOf(elementData)).floatValue();
         cb.setFloat(elementName,fvalue);
      }
      catch(NumberFormatException nfe)
      {
         cb.setString(elementName,elementData.trim());
      }
   }

}

public void characters(char[] ch, int start, int length)
throws SAXException
```

```
   {
      String s = new String(ch, start, length);
      System.out.println("Data: " + s);
      if (!(s.trim().equals("")))
        elementData = elementData + s;
   }

   public void ignorableWhitespace(char[] ch, int start, int length)
   throws SAXException
   {
   }

   public void skippedEntity(String name) throws SAXException
   {
       System.out.println("Skipped entity " + name);
   }
}               //End of XMLContentHandler class

/*
 * Module Name: XMLErrorHandler.java
 * Copyright  (C): IBM Corp. 2001
 * Author:
 * Modification History:
 *
 */

public class XMLErrorHandler implements ErrorHandler
{
   public void warning(SAXParseException pe) throws SAXException
   {
      System.out.println("**Warning**\n" +
                         "  Line:  " + pe.getLineNumber() + "\n" +
                         "  URI:  " + pe.getSystemId() + "\n" +
                         "  Message:  " + pe.getMessage());
      throw new SAXException("Warning encountered");
   }

   public void error(SAXParseException pe) throws SAXException
   {
      System.out.println("**Error**\n" +
                         "  Line:  " + pe.getLineNumber() + "\n" +
                         "  URI:  " + pe.getSystemId() + "\n" +
                         "  Message:  " + pe.getMessage());
      throw new SAXException("Error encountered");

   }

   public void fatalError(SAXParseException pe) throws SAXException
   {
      System.out.println("**Fatal Error**\n" +
                         "  Line:  " + pe.getLineNumber() + "\n" +
                         "  URI:  " + pe.getSystemId() + "\n" +
                         "  Message:  " + pe.getMessage());
      throw new SAXException("Fatal Error encountered");

   }
}               //End of XMLErrorHandler class

}               //End of XMLContentAdapter class
```

Everyplace Intelligent Notification Services Gateway Adapter sample code - Gateway Interface

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors;

import com.ibm.pvc.we.ins.und.server.dispatcher.Msg;
import com.ibm.pvc.we.ins.und.server.dispatcher.DeviceAddress;
import com.ibm.pvc.we.ins.GeneralConstants;

// for extended NS functionality, such as timeout and
// queuing, the Gateway interface will need to be appended
// to:  could use introspective methods such as: 'public boolean canQueue();'

/**
 * interface allowing new Gateway objects to be plugged
 * into the notification server. The Gateway is used
 * to send a message to a device.
 */
public interface Gateway
{
        private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;

        /**
        * This is an alternative success code that is put in for future
        * implementation. All delivery so far is unconfirmed
        */
        public static final int DELIVERY_CONFIRMED=1;

        /**
         * An error occured whilst trying to send the message to corresponding
         * server
         * example: an attempt is made to deliver an Instant Message server.
         * Delivery fails because the server is not responding.
         */
        public static final int SEND_ERROR=-1;

        /**
         * The message has been sent to a server, and no further information
         * is provided.
         * example: message has been sent to an e-mail server, and we do not if
         * subsequent message delivery failed, for example because the email
         * address was incorrect.
         */
        public static final int SENT_TO_SERVER= 0;
        /**
         * Send the message to the recipient whose address
         * has been provided
         *
         * @param msg    The message to be sent by the gateway
               * @return 0 if message sent out successfully
               */
              public int sendMessage(DeviceAddress recipientAddress, Msg msg);

        /**
         * Method Description:
         * format message will format the message by  invoke a transcoder
         */
              void formatMessage(DeviceAddress addr,Msg notificationMsg);
}
```

Everyplace Intelligent Notification Services Gateway Adapter sample code - PAPUser superclass

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors;

import java.net.*;
import com.ibm.wireless.push.*;

import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.GeneralConstants;

/**
 * with the adaptors a.k.a. gateways, we wish to
 * maximize throughput and manage multiple adaptor instances.
 *
 * At the moment each Notification dispatching thread has
 * its own adaptor for each message type. It's initialized
 * once when the thread starts up. Here for example a ServerSocket
 * will be set up to listen for PAP Gateway confirmations of delivery
 *
 * The PAPUser class is used to centralize the initialization
 * of PAP users
 */
public class PAPUser implements com.ibm.logging.IRecordType
{
        private final static String
                insIBMCopyright=GeneralConstants.insIBMCopyright;

        private static Pusher psh=null;
        private static String pushProxyGateway=null;
        private static int listeningPort=-1;
        public static long ADAPTORS = 1 << 37;


        static {
                statinit();
        }

        protected PAPUser(){
        }

        /**
         * static initializer to establish common configuration
         * parameters for adaptors using the Wireless Gateway
         */
```

```
              private static void statinit(){
                        java.util.Properties p= NS_Configuration.getProperties();
                        pushProxyGateway= (String)p.getProperty

(NS_Configuration.ConfigDescriptors.PUSH_PROXY_GATEWAY, "");

                        if (pushProxyGateway=="") {
                                  if (Logging._isLogging) {
                                  Logging._logger.text(ADAPTORS,"PAPUser","statinit",
                        "!! cannot use wireless services as no push proxy url is
specified. ");
                            }
                        }

                        String tmp=(String) p.getProperty
                                (NS_Configuration.ConfigDescriptors.PUSH_PROXY_PORT,
"28292");
                        try {
                                  listeningPort= (Integer.valueOf( tmp)).intValue();
                        }
            catch ( NumberFormatException nfe) {
                                  listeningPort=28242;
                        }

                        try {
                if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,"PAPUser","statinit",
                                                    "DEBUG: creating pusher with url
and no SSL ");
                            }
                                  URL myURL=new URL(pushProxyGateway);

                                  if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,"PAPUser","statinit",
                                                    "DEBUG: URL IS: "+
myURL.toString());
                            }
                                  psh= new Pusher(myURL, null);

                         }
             catch ( Exception e ) {
                if (Logging._isLogging) {

Logging._logger.exception(TYPE_ERROR,"PAPUser","statinit",e);
                                  }
                          }
                }

        /**
         * sends message to Push Proxy Gateway, specifying that
         * bearer and Network can be 'DeliveryQoS.ANY'.
         *
         * @param mess
         * @param addr
         * @return
         */
        int sendMessage(PushMessage mess, PushAddress addr) throws
                PushInvalidArgumentException, PushOperationException, PushIOException
{
```

```
                    // Jlog
                    if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,this,"sendMessage","Entry");
                          }

                    PushResponse res = psh.push(mess, addr);
                    if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,this,"sendMessage",
                                                       "push response:
\n"+res.toString());
                          }

                    if (res.SUCCESS != res.getStatusClass())
                    {
                    if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,this,"sendMessage",
                                                             "Debug: failed");
                                }
                                return Gateway.SEND_ERROR;
                    }

              if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,this,"sendMessage",
                                                             "Debug: success");
                          }
                    return Gateway.SENT_TO_SERVER;
          }

        /**
               * temporary place holder whilst moving to new Zurich API
               *
               * @return
               */
                protected static String getHost(){
                          if (psh==null) return null;
                          return psh.getURL().getHost();
                }

                }
```

Everyplace Intelligent Notification Services Gateway Adapter sample code - Wireless Application Protocol

```
/****************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)              *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved          *
 *                                                                          *
 *  Licensed Materials - Property of IBM                                    *
 *                                                                          *
 *  US Government Users Restricted Rights - Use, duplication or disclosure  *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.           *
 ****************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors;

import java.io.*;
import java.util.*;

import com.ibm.pvc.we.ins.und.server.adaptors.*;
import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;

import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import com.ibm.wireless.push.*;

public class WAP extends PAPUser implements Gateway, com.ibm.logging.IRecordType
{
        private final static String
                insIBMCopyright=GeneralConstants.insIBMCopyright;

        private Transcoder currenttc;
        private DeviceAddress deviceAddr;
        public static Properties gwconfig;
        private String msgbody;
        public static long ADAPTORS = 1 << 37;

        public WAP(){
                super();
                init();
        }

        /**
         * Insert the method's description here.
         * Creation date: (1/19/2001 2:29:13 PM)
         * @return int
         */
        public int sendMessage(DeviceAddress addr,Msg msg) {

                if (Logging._isTracing) {
                                Logging._tracer.text(ADAPTORS,this,"sendMessage",
                                                        "Entry");
                        }
                // initialize variable for this session
                deviceAddr = addr;
                currenttc = null;
                msgbody=null;

                formatMessage(addr,msg);

                if (msgbody != null )
                {
                int returnCode=send();
                return returnCode;
                }
                return SEND_ERROR;
        }

        /* * Insert the method's description here.
         * load transcoder if it has not been loaded yet
         * Creation date: (1/10/2001 3:20:21 PM)
         * @param notificationMsg com.ibm.pcds.ns.server.src.Msg
         */
        public void formatMessage(DeviceAddress deviceAddr, Msg notificationMsg)
        {
                if (Logging._isTracing) {
                               Logging._tracer.text(ADAPTORS,this,"formatMessage",
                                                          "Entry");
                        }

                // get proper transcoder from tc_collection
                currenttc=TcFactory.createTc(deviceAddr,"WAP_TC");

                if (currenttc != null)
                    msgbody=currenttc.format(deviceAddr,notificationMsg);

        }

        /** Insert the method's description here.
         * This method should load gateway configuration file.
         * Creation date: (1/15/2001 3:51:37 PM)
         */
        public void init()
        {
                if ( gwconfig == null ) {
                         gwconfig= NS_Configuration.getProperties();
                        }

                }

        public int send() {
                if (Logging._isTracing) {
                                Logging._tracer.text(ADAPTORS,this,"send","Entry");
                        }
                int returnCode=SEND_ERROR;


                        String message; // message as strung together from the NS
Msg.
                        PushMessage p_message;
                        PushAddress p_address;
                        String _phoneNumber = deviceAddr.getAddress();

                          try {
                                p_message=new PushMessage();
                                p_message.setContent( msgbody, "text/vnd.wap.wml");
                           p_message.setDeliveryMethod(
DeliveryQoS.DELIVERY_UNCONFIRMED);
                                p_address= new PushAddress();
                                // "PLMN" replaced by IPv4 for testing: we have
                                        // no WAP enabled phones.
                                // PLMN== public land mobile network
                                p_address.setAddress( _phoneNumber, "IPv4",
                                        PAPUser.getHost());

                                returnCode= sendMessage( p_message, p_address);
                        } catch ( PushException pe ) {
                                //pe.printStackTrace();
                                // JLog
                                if (Logging._isLogging) {
                                      Logging._logger.exception(TYPE_ERROR,this,
                                                          "send",pe);
                                }
                                return SEND_ERROR;
                        }

                return returnCode;

                }
}
```

Everyplace Intelligent Notification Services Gateway Adapter sample code - Instant Messaging / Sametime

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors;

import com.ubique.vp.types.*;
import com.ubique.vp.VpSession;
import com.ubique.vp.services.*;
import com.ubique.vp.constants.VpkError;

import java.util.*;

import com.ibm.pvc.we.ins.und.server.dispatcher.*;

import com.ibm.pvc.we.ins.und.server.adaptors.*;
import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;
import com.ibm.pvc.we.ins.GeneralConstants;

/**
 * Adaptor for sending out messages to a Sametime Server
 *   -e.g. messaging.ibm.com.
 * Due to the way that the Sametime1.5 client code is written
 * only one Server can be logged onto per JVM
 *
 * A request to send a message is wrapped as a MessageJob object,
 * and queued for processing. Processing a job involves communicating
 * with the Sametime server and generating a number of events. Events
 * are wrapped in a functor object, and processed by the
 * Worker.
 *
 */
public class SametimeIMGW  implements Gateway, com.ibm.logging.IRecordType
{
        private final static String
                insIBMCopyright=GeneralConstants.insIBMCopyright;

        // variables required to load transcoder
        private Transcoder currenttc;
        static Properties gwconfig;
        private String msgbody;
        static long ADAPTORS=1<<37;  //mask for adaptor jlog support

        private static VpSession sametimeSession;
              private static CommunityService commService;
              private static InstantMessagingService imService;
```

```
      private static String server;
private static String ourClientAddress;
      private static String ourClientPassword;

// map to address to jobs
private static Hashtable mapAddressToJobs;
// keep track of request ids
private static Hashtable mapIntegerToAddress;
// temp book keeping trick
private static Hashtable mapAddressToChannelCreated;
// queue of messaging job requests
private static Vector jobVector=new Vector();
// the message type used in channel creation
private static final int MESSAGE_TYPE= 1;


// processes jobs and events
private static Worker wrk;
// community event listener
private static CommSrvListener clist;
// instant messaging event listener
        private static IMSrvListener imlist;
// whether is logged in. allows checks of status before
private static boolean logged;



/** Method Description:
 * format message will format the message by invoke default transcoder
 */
public void formatMessage(DeviceAddress addr, Msg notificationMsg) {

if (Logging._isTracing) {
                Logging._tracer.text(ADAPTORS,this,"formatMessage",
                                        "Entry");
                }

// get proper transcoder from tc_collection
currenttc=TcFactory.createTc(addr,"IMSAMETIME1V5_TC");

if (currenttc != null)
    msgbody=currenttc.format(addr,notificationMsg);
}


      /**
       * static initialization of our session with Sametime server.
       */
static{
              logged=false;
              wrk=new Worker();
              wrk.start();

              // create VpSession
              sametimeSession=new VpSession();

              // create components instance
              commService=new CommunityService( sametimeSession);
              imService=new InstantMessagingService( sametimeSession);

              // resgister instances to listerner for handling events
```

```
                      clist=new CommSrvListener();
                      imlist= new IMSrvListener();
                      commService.addListener( clist);
                      imService.addListener( imlist);

                      server=
                          NS_Configuration.getProperties().getProperty

(NS_Configuration.ConfigDescriptors.SAMETIME_SERVER_URL).trim();
                      ourClientAddress=
                          NS_Configuration.getProperties().getProperty

(NS_Configuration.ConfigDescriptors.SAMETIME_CLIENT_USERID).trim();
                      ourClientPassword=
                          NS_Configuration.getProperties().getProperty

(NS_Configuration.ConfigDescriptors.SAMETIME_CLIENT_PASSWORD).trim();

                      // update sametime configuration while change has been
                          // made via admin console
                      CCListener cclist=new CCListener();
                      NS_Configuration.addListener( cclist);

                      commService.loginByPassword( server, ourClientAddress,
                          ourClientPassword);
        }

              public SametimeIMGW(){

          // on the first initialization of this class,
                  // we have to wait long enough for login to be performed
                      if (!logged) {
                                synchronized(jobVector){
                                        try {
                                                  jobVector.wait(10000);
                                                          // this works fine with the
server and
                                                          // occurs for first message
only
                                        } catch (InterruptedException ie) {
                                                  if (Logging._isLogging) {

Logging._logger.exception(TYPE_WARN, this,
                                                                 "SametimeIMGW",
ie);
                                                  }
                                        }
                                }       //sychronized
                      } //logged

                // get Properties file
                      if ( gwconfig == null )
                            gwconfig= NS_Configuration.getProperties();

                      //when an instance is created,
                          //make sure referencecount is incremented.
                      instanceCreated();
              }

              public int sendMessage(DeviceAddress addr, Msg msg){

                              if (Logging._isTracing) {
```

```
                       Logging._tracer.message(ADAPTORS,this,"sendMessage",
                           "ENTER_SAMETIME15_SENDMESSAGE");
                }
                      // initialize variable for this session
          currenttc = null;
          msgbody=null;

          formatMessage(addr,msg);

          if (msgbody != null )
          {
              int returnCode=send(addr);
              return returnCode;
          }
          return SEND_ERROR;
       }


      private class ReturnCode {
                private int c;

    public ReturnCode(){
                          c=SEND_ERROR;
                }

    public final int getReturnCode(){
                            return c;
                }

    public final void setReturnCode(int code){
                            c=code;
                }
      }

      /**
       * wraps a messageing request, and hides the notification that
       * we do to release the requesting thread (see sendMessge())
       */
      private class MessageJob {
                private VpkId stid;
                private String myaddress;
                private String myuserid;
                private String mymessage;
                private ReturnCode myReturnCode;
                public MessageJob(){
    }

public ReturnCode defineRequest( String address, String message){

  myaddress=address;
                    mymessage=message;
                    myReturnCode= new ReturnCode();
                    return myReturnCode;
            }

public void setId(VpkId anid){
                    stid= anid;
            }

public VpkId getId(){
                    return stid;
            }
  public String getAddress(){
                    return myaddress;
            }

  public String getMessage(){
                    return mymessage;
            }

  public void setJobCompleted( int returnCode){
                      myReturnCode.setReturnCode( returnCode);
                      synchronized ( myReturnCode){
                                  myReturnCode.notifyAll();
                        }
            }
    }


/**
       * Used to wrap events generated by interaction with server
       */
        private interface Functor {
                  public void execute();
        }


/**
 * Community service event listener
 */
  private static class CommSrvListener implements CommunityServiceListener {
          public void statusUpdated(VpkUserStatus status) {
                  if (Logging._isTracing) {
                          Logging._tracer.message(TYPE_MISC_DATA,this,
                                  "statusUpdated","STATUS_UPDATED");

                  }
          }

          public void loggedOut(Integer reason) {
                  logged=false;
                  if (Logging._isLogging) {
                          Logging._logger.message(TYPE_MISC_DATA,this,
                                  "loggedOut","LOGGED_OUT",
                                  STErrorCodes.getErrorDescription( reason));
                  }

                  // the next line will only be true if the logout call
                          // was requested locally - i..e from the finalizer
                          // setup described above
                  if (reason.intValue()== VpkError.VPK_OK)
                          return; // this was the logout we requested

                  // if logged out, then jar worker thread out of its
                          // present task, whatever that is
                  synchronized( jobVector){
                          wrk.interrupt();
                  }

                  if (reason.intValue()==VpkError.VPK_CONNECT_BAD_LOGIN){
                  // make sure the administrator is properly warned
                          // that there is an incorrect configuration
                  if (Logging._isLogging)
```

```
                                Logging._logger.text(TYPE_ERROR, this,"LOGGED_OUT",
                                        "BAD_LOG_IN:
"+STErrorCodes.getErrorDescription( reason));
                                return;
                        }
                        relogin();
                }

                public void relogin()
                {

                        Thread clock=new Thread(){
                                public void run() {
                                        synchronized (this){
                                                try {
                                                        wait(1000); // arbitrary
number
                                                } catch (InterruptedException ie) {
                                                }
                                        }


                                        commService.loginByPassword( server,
ourClientAddress,
                                                ourClientPassword);

                                }
                        };
                        clock.start();

                }

                public void setPrivacyDenied(Integer reason) {

                }
                public void privacyListUpdated(VpkPrivacyList plist) {
                }
                public void namesResolved(Integer rescode,
                        final VpResolveMatches[] mymatches) {
                        if (Logging._isTracing) {
                                Logging._tracer.message(TYPE_MISC_DATA, this,
                                        "namesResolved",
"SUCCESSFULL_NAME_RESOLUTION");
                        }
                        Functor f=new Functor(){
                                public void execute() {
                                        int lenvector;
                                        int lenmatches= mymatches.length;
                                        String s;
                                        VpResolveInfo[] inf;
                                        VpkId stid;
                                        MessageJob ajob;
                                        Vector v;
                                        Integer channelId;
                                        Object o;
                                        for (int i=0; i<lenmatches; i++) {
                                                inf=mymatches[i].getMatches();
                                                s= mymatches[i].getResolveString();
                                                if (inf==null || inf.length==0) {

v=(Vector)mapAddressToJobs.get( s);
                                                        int len= v.size();
                                                        for (int j=0; j<len; i++) {
                                                                ajob=(MessageJob)
v.elementAt( j);
                                                                ajob.setJobCompleted(
SEND_ERROR);
                                                        }
                                                        mapAddressToJobs.remove( s);
                                                                // user is not on
server,
                                                                //we've been passed
dud address continue;
                                                }
                                                stid= inf[0].getId();
                                                o= mapAddressToChannelCreated.get(
s);
                                                if (o==null) {
                                                        if (Logging._isTracing) {

Logging._tracer.message(TYPE_MISC_DATA,this,
                                                                    "execute",
"CREATING_MESSAGE_CHANNEL");
                                                        }
                                                        channelId=
imService.createMessage( stid,

EncLevel.ENC_LEVEL_NONE, MESSAGE_TYPE);

mapAddressToChannelCreated.put( s, new Object());
                                                        mapIntegerToAddress.put(
channelId, s);
                                                }
                                        }// through matches
                                } // execute
                        };
                        wrk.processFunctor( f);
                }
                public void resolveNamesFailed(Integer code, Integer reason) {
                        // request failed

                        if (Logging._isTracing) {
                                Logging._tracer.message(TYPE_MISC_DATA,this,
                                        "resolveNamesFailed","RESOLUTION_FAILED",
                                        STErrorCodes.getErrorDescription(reason));
                        }

                        Functor f=new Functor(){
                                public void execute() {
                                        Collection c= mapAddressToJobs.values();
                                        Iterator iter=c.iterator();
                                        Vector v;
                                        int len;
                                        MessageJob mj;
                                        while (iter.hasNext()) {
                                                v= (Vector)iter.next();
                                                len=v.size();
                                                for (int i=0; i<len; i++) {
                                                        mj=(MessageJob) v.elementAt(
i);
                                                        mj.setJobCompleted(
SEND_ERROR);
                                                }
                                        }
                                }
                        };
                }

                // if log in is successful
                public void loggedIn() {
                        logged=true;
                        if (Logging._isLogging) {
                                Logging._logger.message(TYPE_INFO,this,
                                    "loggedIn","SAMETIME_LOGGED_IN");
                        }
                }
                public void adminMessage(String s) {
                        if (Logging._isLogging) {
                                Logging._logger.message(TYPE_INFO,this,
                                        "adminMessge","SAMETIME_ADMIN_MESSAGE", s);
                        }
                }
        }

        /**
         * Our messageing event listener
         */
        private static class IMSrvListener implements
                InstantMessagingServiceListener {
                public void messageDestroyed(Integer code, Integer reason) {
                        if (Logging._isTracing) {

Logging._tracer.message(TYPE_MISC_DATA+Debug.ADAPTORS,this,
                                    "messageDestroyed","SAMETIME_MESSAGE_DESTROYED",
                                        STErrorCodes.getErrorDescription( reason));
                        }
                }
                public void messageDenied(final Integer code, Integer reason) {
                        if (Logging._isTracing) {
                                Logging._tracer.message(TYPE_MISC_DATA,this,
                                        "messageDenied","SAMETIME_MESSAGE_DENIED",
                                        STErrorCodes.getErrorDescription( reason));
                        }
                        Functor f=new Functor(){
                                public void execute() {
                                        String s=(String)mapIntegerToAddress.get(
code);

                                        Vector v=(Vector)mapAddressToJobs.remove( s);
                                        int len=v.size();
                                        MessageJob mj;
                                        for (int i=0; i<len; i++) {
                                                mj= (MessageJob) v.elementAt( i);
                                                mj.setJobCompleted( SEND_ERROR);
                                        }
                                }
                        };
                        wrk.processFunctor( f);
                }
                public void messageCreated(final Integer code, EncLevel encLevel,
                        int type, VpkLoginInfo loginInfo,final VpkUserStatus status,
                        boolean isOriginator) {
                        if (Logging._isTracing) {
                                Logging._tracer.message(TYPE_MISC_DATA,this,
                                        "messageCreated","SAMETIME_MESSAGE_CREATED");
                        }
                        // if we aren't originator, just let other party close
```

```
channel
                        if (isOriginator) {
                                Functor f=new Functor(){
                                        public void execute() {
                                                String s;
                                                StringBuffer text=new
StringBuffer("Notification Server
                                                        Message:\n");
                                                MessageJob ajob;
                                                Vector v;
                                                int len;
                                                // make a check on status?
                                                s=(String)mapIntegerToAddress.get(
code);
                                                v=(Vector)mapAddressToJobs.get( s);
                                                len=v.size();
                                                for (int i=0; i<len; i++) {
                                                        ajob=
(MessageJob)v.elementAt( i);
                                                        text.append(
ajob.getMessage());
                                                        ajob.setJobCompleted(
SENT_TO_SERVER);
                                                }
                                                imService.sendText( code,
text.toString());
                                                imService.destroyMessage( code,
                                                        new
Integer(VpkError.VPK_OK));
                                                mapAddressToJobs.remove( s);
                                        }
                                };
                                wrk.processFunctor( f);
                        }
                }
                public void textReceived(Integer code, String s) {

                }
                public void dataReceived(Integer code, int i, int j, byte[] b) {
                }
        }
        /**
         * The Worker allows us a thread separate from the Sametime 1.5
         * toolkit threads. Once the worker has been notified of one
         * or more message jobs on its queue, it sends out a resolution request
         * to the sametime server. A number of messaging events are created
         * and processed by the worker. The worker only processes
         * events originating locally.
         */
        private static class Worker extends Thread {
                private MessageJob[] myjobs;
                private Vector myFunctors=new Vector();
                private boolean active=true;

                public void pleaseStop(){
                        active=false;
                }

                public void processFunctor(Functor f){
                        myFunctors.addElement( f);
                        synchronized (myFunctors){
                                myFunctors.notifyAll();
                        }
                }

                public void run() {
                        int numJobs=0;

                        String s;
                        String[] resolveList;
                        int numEvents=0;
                        Functor f;
                        while (active) {
                                try {
                                        synchronized ( jobVector){
                                                if (jobVector.size()==0)
jobVector.wait();
                                                mapAddressToJobs=new Hashtable(100);
                                                mapIntegerToAddress=new
Hashtable(100);
                                                mapAddressToChannelCreated=new
Hashtable(100);
                                                numJobs= jobVector.size();
                                                myjobs= new MessageJob[ numJobs];
                                                resolveList=new String[ numJobs];
                                                        // may have multiple mentions
of the same string

                                                // create address-jobvector hashtable
                                                // create address array
                                                for (int i=numJobs-1 ; i>-1; i--) {

myjobs[i]=(MessageJob)jobVector.elementAt( i);
                                                        jobVector.removeElementAt(
i);

                                                        s= myjobs[i].getAddress();
                                                        resolveList[i]=s;
                                                        Vector
v=(Vector)mapAddressToJobs.get( s);
                                                        if (v==null) v=new Vector();
                                                        v.addElement( myjobs[i]);
                                                        mapAddressToJobs.put( s, v);
                                                }
                                        }
                                        commService.resolveNames( resolveList, true,
true,
                                                true, false);

                                        // mapAddressToJobs is pruned for dud
addresses
                                        while ( mapAddressToJobs.size() != 0) {
                                                // still have addresses to be acted
on
                                                synchronized ( myFunctors){
                                                        if (myFunctors.size()==0)
myFunctors.wait();
                                                }
                                                // process events in order of arrival
                                                f=(Functor)myFunctors.elementAt( 0);
                                                myFunctors.removeElementAt( 0);
                                                f.execute();
                                        }   // while events
                                } catch (Exception e) {
                                        if (Logging._isTracing) {

                                                        Logging._tracer.exception

(TYPE_MISC_DATA+Debug.ADAPTORS, this, "run",e);
                                        }
                                }
                        }  // while active
                }  // run
        }

        public int send(DeviceAddress addr) {
                        if (Logging._isTracing) {
                        Logging._tracer.text(ADAPTORS,this,"send","Entry");
                }

                MessageJob j= new MessageJob();

                String srvrDst;
                String addrsDst;
                addrsDst=addr.getAddress();
                srvrDst=addr.getServer();

                // check if has logged on to sametime server
                if (!commService.isLoggedIn()) {
                        if (Logging._isLogging) {
                        Logging._logger.message(TYPE_WARN,this,"sendMessage",
                    "NOT_LOGGED_WHILE_SENDING_MESSAGE");
                        }
                        return SEND_ERROR;
                }

                if ( srvrDst==null || !srvrDst.equals( server)) {
                        if (Logging._isTracing) {
                        Logging._tracer.message(TYPE_MISC_DATA,this,"sendMessage",
                            "INCOMPATIBLE_SERVER_COMMUNITY");
                        }
                        return SEND_ERROR;
                }
                // This is where the request gets its final format, at the moment
                ReturnCode c=j.defineRequest(addrsDst,msgbody);

                jobVector.addElement( j);

                synchronized (jobVector){
                        jobVector.notifyAll();
                }

                try {
                        synchronized ( c){ // block on return code until return code
set
                        c.wait(10000);  // should this time out?
                                //( scenario in which server fails to respond?)

                        }
                } catch ( InterruptedException ie) {
                        if (Logging._isLogging) {
                                Logging._logger.message(TYPE_WARN, this,
"sendMessage",
                                        ie.toString());
                        }
                }
                return c.getReturnCode();
        }

        private static int referenceCount=0;
                // create a new static variable to count instances

        private static class CCListener implements ConfigurationChangeEventListener
        {
                // update sametime configuation
                public void configurationChanged()
                {

                String new_server=
                 NS_Configuration.getProperties().getProperty
                (NS_Configuration.ConfigDescriptors.SAMETIME_SERVER_URL).trim();
                String new_ourClientAddress=
                NS_Configuration.getProperties().getProperty
        (NS_Configuration.ConfigDescriptors.SAMETIME_CLIENT_USERID).trim();
                String new_ourClientPassword=
                  NS_Configuration.getProperties().getProperty

(NS_Configuration.ConfigDescriptors.SAMETIME_CLIENT_PASSWORD).trim();

                if ( !(server.equals(new_server) &&
                        ourClientAddress.equals(new_ourClientAddress) &&
                        ourClientPassword.equals(new_ourClientPassword)))
                {

                        server = new_server;
                        ourClientAddress=new_ourClientAddress;
                        ourClientPassword=new_ourClientPassword;

                        // make sure log off before log in by using new configuration
                        if (commService.isLoggedIn())
                                commService.logout();
                        clist.relogin();
                }
                }
        }

        // make sure that every time an instance is garbage collected,
                // instanceDestroyed is called:
        protected void finalize(){
                instanceDestroyed();
                }

        // create a pair of static synchronized methods to count
                // the references as they are created
        private static synchronized void instanceCreated(){
        referenceCount=referenceCount+1;
        }

        private static  synchronized void instanceDestroyed(){
        referenceCount=referenceCount-1;
        if (referenceCount==0){
                // if there are no more references, do a statically
                // synchronized call on logout
                commService.logout();
         wrk.pleaseStop();         // stop the worker thread
         synchronized( jobVector)
                                {
```

```
                  jobVector.notifyAll();
          // make sure that the worker thread isn't waiting on its queue
                  }
          }
  }
}
```

Everyplace Intelligent Notification Services Gateway Adapter sample code - Short Messaging Service

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors;

import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;
import com.ibm.pvc.we.ins.GeneralConstants;
import java.util.*;
import com.ibm.wireless.push.*;

/**
* uses the wireless gateway to send out an SMS using the push
* API
*/
public class SMSPush extends PAPUser implements Gateway, com.ibm.logging.IRecordType
{
        private final static String
                insIBMCopyright=GeneralConstants.insIBMCopyright;
        public static long ADAPTORS = 1 << 37;
        private Transcoder currenttc;
        private DeviceAddress deviceAddr;
        static Properties gwconfig;
        private String msgbody;


        /** Method Description:
         * format message will format the message by either invoke default
                 * transcoder or apply user style sheet
         */
        public void formatMessage(DeviceAddress addr, Msg notificationMsg)
        {
                // JLog
                if (Logging._isTracing) {
                             Logging._tracer.text(ADAPTORS,this,"formatMessage",
                                                        "Entry");
                }
        // get proper transcoder from tc_collection
        currenttc=TcFactory.createTc(deviceAddr,"SMS_TC");

        if (currenttc != null)
            msgbody=currenttc.format(deviceAddr,notificationMsg);
        }
```

```
  public int send()
  {
        if (Logging._isTracing)
        {
                Logging._tracer.text(ADAPTORS, this, "send", "Entry");
        }
        int returnCode = -1;
  PushMessage p_message;
  PushAddress p_address;

  // check if formated message is still too long to fit the device
  if (msgbody.startsWith("!"))
  {

          if (Logging._isLogging)
          {
                  Logging._logger.message(TYPE_WARN, this, "send",
                          "MSG_TOOLONG_FOR_SMS ");
          }
          return -1;
  }

  try
  {
          // create simple SMS message
          p_message = new PushMessage();
          p_message.setContent(msgbody, "text/plain");
          p_address = new PushAddress();


          //"PLMN" public land mobile network
          p_address.setAddress(deviceAddr.getAddress(), PushAddress.PLMN,
                  PAPUser.getHost());
          p_address.addExtension("delivery", "sms");


          returnCode = sendMessage(p_message, p_address);
  }
  catch (PushException pe)
  {

          // JLog
          if (Logging._isLogging)
          {
                  Logging._logger.exception(TYPE_ERROR, this, "send", pe);
          }
          return -1;
  }
  return returnCode;

}



public SMSPush() {
          super();
          init();
  }

/**
 * Send the message to the recipient whose address
```

```
       * has been provided, via SMS, to Wireless Gateway
       * configured in PAPUser
       *
       * @param devAdd
       * @param ns_message
       * @return 0 if message sent out successfully
       */
      public int sendMessage(DeviceAddress addr, Msg msg) {

                if (Logging._isTracing) {
                        Logging._tracer.text(ADAPTORS,this,"sendMessage","Entry");
                }

                int returnCode=-1;

                // initialize variable for this session
            deviceAddr = addr;
            currenttc = null;
            msgbody=null;

            formatMessage(addr,msg);

            if (msgbody != null )
        {
            returnCode=send();
            return returnCode;
        }
        return SEND_ERROR;
        }



            public void init()
      {
          if ( gwconfig == null ) {
            gwconfig= NS_Configuration.getProperties();
          }
       }
}
```

Everyplace Intelligent Notification Services Gateway Adapter sample code - SMTP Email

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors;

import com.ibm.caf.mail.*;

import java.lang.*;
import java.util.*;


import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;
import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.GeneralConstants;

/**
 * Adaptor send notification message to a SMTP server.
 */

public class Mail implements Gateway, com.ibm.logging.IRecordType
{
   private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;

   private String emailAddress;
   private SendMail mailSender;
   private int port;
   private String emailServer, transportProtocol;

   private Transcoder currenttc;
   private DeviceAddress deviceAddr;
   private Msg msg;
   private String msgbody;

   private Properties gwconfig;
   static long ADAPTORS=1<<37;

   /** Method Description:
     * format message will format the message by either invoke
         * default transcoder or apply user style sheet
     */
    public void formatMessage(DeviceAddress addr, Msg notificationMsg) {
         if (Logging._isTracing) {
                        Logging._tracer.text

(ADAPTORS,this,"formatMessage","Entry");
                }
```

```
        // get proper transcoder from tc_collection
        currenttc=TcFactory.createTc(deviceAddr,"EMAIL_TC");

              if (currenttc != null)
              msgbody=currenttc.format(deviceAddr,notificationMsg);
      }


   /** Method Description:
     * Send message to recipient's email address
     * return SEND_ERROR or SENT_TO_SERVER
     */
    public int send() {
                    if (Logging._isTracing) {
                            Logging._tracer.text(ADAPTORS,this,"send","Entry");
                  }

                    emailAddress=deviceAddr.getAddress();

                    try {

                          //formUserid - the sender address
                          //             may need to be replaced by user's email
address
                          //to - the recepient address
                          //subj - the subject of the message
                          //body - the body of the message
                          MessageData eMail=new MessageData( msg.fromUserid,

emailAddress,

msg.subject,

msgbody);

                          mailSender.send( eMail);
                    }
        catch ( Exception e ) {
                          if (Logging._isLogging) {
                              Logging._logger.exception(TYPE_ERROR,this,"send",e);
                            }
                    return SEND_ERROR;
                    }
                return SENT_TO_SERVER;
    }

   /** Method Description:
           * Get emailserver host and port information from configuration
           */
    public Mail()
          {
                    // get email host from config object
                    emailServer= (String)NS_Configuration.getProperties().get
                        (NS_Configuration.ConfigDescriptors.EMAIL_HOST);

        // get port String from config object, and turn it into an int (via an
Integer)
                    String str_port=(String) NS_Configuration.getProperties().get
                (NS_Configuration.ConfigDescriptors.EMAIL_SERVER_PORT);
                    port= (Integer.valueOf( str_port) ).intValue();

                    transportProtocol= (String)NS_Configuration.getProperties().get
```

```
                        (NS_Configuration.ConfigDescriptors.EMAIL_PROTOCOL);

                    try {
                          // create SendMailJavaBean
               mailSender= new SendMailJavaBean( transportProtocol,

emailServer,

port,

null,

null);
                    }
        catch ( Exception e ) {
                        if (Logging._isLogging) {
                    Logging._logger.exception(TYPE_ERROR,this,"Mail",e);
                    }
        }

                    if ( gwconfig == null ) {
                        gwconfig= NS_Configuration.getProperties();
                    }
          }

          /**
           * Formate message and send
           *
           * @param addr   email address
           * @param msg    Msg object encapsulating message
           * @return success code
           */
          public int sendMessage(DeviceAddress addr, Msg msg)
          {
                    if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,this,"sendMessage","Entry");
                    }

              // initialize variable for this session
              deviceAddr = addr;
              currenttc = null;
              msgbody=null;
              this.msg=msg;

              formatMessage(addr,msg);
              int returnCode=send();
              return returnCode;
    }
}
```

Everyplace Intelligent Notification Services Gateway Adapter sample code - Email Push

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
  package com.ibm.pvc.we.ins.und.server.adaptors;

  import com.ibm.wireless.push.*;
  import java.io.*;
  import java.util.*;

  import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;
  import com.ibm.pvc.we.ins.und.server.dispatcher.*;
  import com.ibm.pvc.we.ins.GeneralConstants;

  /**
   * Insert the type's description here.
   * Adaptor send notification message to IBM Push Proxy Gateway.
   * Then message will be forwarded to a SMTP server.
   */
  public class MailPush extends PAPUser implements Gateway,
  com.ibm.logging.IRecordType
  {
          private final static String
                  insIBMCopyright=GeneralConstants.insIBMCopyright;

          // variables per instance
          private Transcoder currenttc;
          private DeviceAddress deviceAddr;
          private String msgbody;

          public static Properties gwconfig;
          public static long ADAPTORS=1<<37;

          /**
           * MailPush constructor comment.
           */
          public MailPush()
          {
            if ( gwconfig == null ) {
                   gwconfig= NS_Configuration.getProperties();
                  }
          }

          /** Insert the method's description here.
           * load transcoder and save transformed message in msgbody
           * @param notificationMsg com.ibm.pcds.ns.server.src.Msg
           */
          public void formatMessage(DeviceAddress deviceAddr, Msg notificationMsg)
          {
                if (Logging._isTracing) {
                                Logging._tracer.text(ADAPTORS,this,"formatMessage",
                                        "Entry");
                        }

                // get proper transcoder from tc_collection
                currenttc=TcFactory.createTc("MAILPUSH_TC");

            // transform message and save in msgbody
                if (currenttc != null)
                    msgbody=currenttc.format(deviceAddr,notificationMsg);
         }

        /** Insert the method's description here.
         * Send message to recipient's email address
         */
        public int send(Msg msg)
        {
              if (Logging._isTracing) {
                                Logging._tracer.text(ADAPTORS,this,"send","Entry");
                        }
              int returnCode=SEND_ERROR;

                    String message;
                    PushMessage p_message;
                    PushAddress p_address;

         // get recipient's email address
                    String _emailAddr = deviceAddr.getAddress();

                    try {

                  // create push message
                            p_message=new PushMessage();

                            //The content type must be a text MIME media type,
                                //other types are rejected.
                            p_message.setContent( msgbody, "text/plain");

              // fromUserid may need to be replaced by user's email address
                            p_message.addContentHeader(SMTPHeader.FROM,
msg.fromUserid);
                            p_message.addContentHeader(SMTPHeader.SUBJECT,
msg.subject);

                  // create push address
                            p_address= new PushAddress();

              //address - the address of the mobile device.
                            //type - the address type.
                            //domain - the domain name of the messaging gateway.
                            p_address.setAddress( _emailAddr, PushAddress.SMTP,
                                PAPUser.getHost());

              // send message to push proxy gateway
                            returnCode= sendMessage( p_message, p_address);
                            }
                    catch ( PushInvalidArgumentException piae) {
                                  if (Logging._isLogging) {
                            Logging._logger.exception(TYPE_ERROR,this,"send",piae);
                                 }
                        }
                    catch ( PushOperationException poe ) {
                                                        if (Logging._isLogging) {
                              Logging._logger.exception(TYPE_ERROR,this,"send",poe);
                                  }
                  }
                    catch ( PushIOException pioe ) {
                                                if (Logging._isLogging) {
                              Logging._logger.exception(TYPE_ERROR,this,"send",pioe);
                                        }
                        }
              return returnCode;
         }


      /**
       * Insert the method's description here.
       * Format message then send it to recipient.
       * Return send error if format or send is not sucessful.
       * @return int
       */
      public int sendMessage(DeviceAddress addr,Msg msg) {

              if (Logging._isTracing) {

Logging._tracer.text(ADAPTORS,this,"sendMessage","Entry");
                    }

              // initialize variable for this session
              deviceAddr = addr;
              currenttc = null;
              msgbody=null;

              formatMessage(addr,msg);

              if (msgbody != null )
              {
                int returnCode=send(msg);
                return returnCode;
              }
              return SEND_ERROR;
      }
  }
```

Everyplace Intelligent Notification Services Transcoder sample code - Transcoder Interface

```
 /*****************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors.transcoder;

import java.util.Properties;

import com.ibm.pvc.we.ins.und.server.dispatcher.Msg;
import com.ibm.pvc.we.ins.und.server.adaptors.*;
import com.ibm.pvc.we.ins.und.server.dispatcher.DeviceAddress;

public interface Transcoder
{

   /**
    * Format message by either invoke default formatting method or use a stylesheet
    * @param msg    The message to be sent by the gateway
        * @param addr   Recipient's device address information
    * @return       Formatted message will be returned as a String
    */
   public String format(DeviceAddress addr,Msg msg);

   /**
    * Pass stylesheet configuration information to transcoder
    * @param gwconfig    Properties object which contains stylesheet
        * configuration information
        * @return
    */
   public void setGWConfigFile(Properties gwconfig);

}
```

Everyplace Intelligent Notification Services Transcoder sample code - *TcFactory.java*

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors.transcoder;

import java.util.*;

import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.GeneralConstants;

/**
 * Insert the type's description here.
 * This class helps adaptor to create transcoder instance upon request.
 * It could load transcoder class specified by configuration and
 * return the proper transcoder each time a send message request is initiated
 */

public class TcFactory implements com.ibm.logging.IRecordType
{
   private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;
   public static long TRANSCODERS=1<<38;


        /**
         * Insert the method's description here.
         * If str == null, return "" instead.
         * If str != null, return uppercase with version modification.
         * @return java.lang.String
         */
        public static String initString(String str) {
                String temp;
                if (str == null)
                return "";
                else
                  temp= str.toUpperCase();

                // modify  version expression  eg. replace 1.5 with 1V5
                  temp = temp.replace('.','V');
                  return temp;
        }

        /**
         * Insert the method's description here.
         * If the key could be found in configuration, load,initialize and return.
         * Otherwise return null.
         * Creation date: (1/25/2001 9:57:57 AM)
         */
```

```
public static Transcoder loadTc(String key)
{
   if (Logging._isTracing) {
      Logging._tracer.text(TRANSCODERS,
                        "TcFactory","loadTc","Entry");
                                  }
        Transcoder tc=null;

        // load transcoder from properties file
        Properties  gwconfig=NS_Configuration.getProperties();
        String tcvalue=gwconfig.getProperty(key);
        if ( tcvalue != null )
        {
        try
                {
               Class c = Class.forName(tcvalue);
               tc = (Transcoder) c.newInstance();
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                        // JLog
                        if (Logging._isLogging) {
                    Logging._logger.message(TYPE_ERROR,
                                "TcFactory","loadTc","TC_LOAD_ERROR");
                         Logging._logger.exception(TYPE_ERROR,
                                "TcFactory","loadTc",e);
                          }
                        return tc;
                }

                // set TC properties
                tc.setGWConfigFile(gwconfig);
   }
        return tc;
}

/**
 * Insert the method's description here.
 *
 * The configuration associates keys with Transcoder classnames.
 * Each key has a device descriptor and a suffix -- "_TC".
 * The descriptors are composed in the format of DevicetypePrototolVersion.
 * Following are rules applied to the naming
         * convention of device descriptors.
 * @The device descriptor must minimally define a device type.
 * @All values are converted to uppercase when constructing a key
 * @If a device descriptor has no subtype, then no placeholder
         * is used, e.g. WAP1V2
 * Example:    IMSAMETIME1V5_TC
 *                         WAP_TC
 *             EMAIL_TC
 *
 * The factory compose keys according to descriptor information
 * stored in DeviceAddress and compares it to the
 * configuration keys. If one is an exact match, then
 * corresponding transcoder is loaded. If not, a partial match is
 * checked for, first DevicetypeProtocol, then just Devicetype.
 * The checks against most precise device descriptor to most general.
 * The 'best' matching class is instantiated if possible, then
 * returned.
 * If no match could be found, a default transcoder will be returned.
```

```
         * A default key will be checked against configuration. If a match is
         * is found, a default transcoder instance will be returned. Otherwise
         * null will be returned.
         *
         * Creation date: (2/6/2001 2:41:07 PM)
         */

         public static Transcoder createTc(DeviceAddress deviceAddr,
                 String defaultKey) {
                if (Logging._isTracing) {
                        Logging._tracer.text(TRANSCODERS,
                                "TcFactory","createTc(DeviceAddress,Key)","Entry");
                                          }
                Transcoder currenttc=null;


                Transcoder defaulttc=loadTc(defaultKey);
                if ( defaulttc == null)
                {
                         if (Logging._isLogging) {
                             Logging._logger.message(TYPE_ERROR,
                                        "TcFactory","createTc(DeviceAddress,Key)",
                                        "NEED_DEFAULT_TC",defaultKey);
                                          }
                        }

                String key= new String();

                String devicetype=initString(deviceAddr.getDeviceType());
                String protocol=initString(deviceAddr.getProtocol());
                String version=initString(deviceAddr.getVersion());

                key = new

String(devicetype).concat(protocol).concat(version).concat("_TC");

                // devicetype_protocol_version transcoder could not be found,
                        // check devicetype_protocol transcoder
                if ((currenttc=loadTc(key)) == null)
                   {
                           key = new
String(devicetype).concat(protocol).concat("_TC");
                           if ((currenttc=loadTc(key)) == null)
                           {
                                   key = new String(devicetype).concat("_TC");
                                   if ((currenttc=loadTc(key))== null)
                                   {
                                           // default tc will be returned if no
transcoder
                                                   // could be found in configuration
                                           if (Logging._isTracing) {
                                   Logging._tracer.text(TRANSCODERS,

"TcFactory","createTc(DeviceAddress,Key)",
"Default transcoder for this device type will be returned. \n Key for default TC is :
{0} ",
                                                defaultKey);
                                          }
                                           currenttc=defaulttc;
                                   }
                           }
                   }
```

```
		return currenttc;
	}


	/**
	 * Insert the method's description here.
	 * Load transcoder from configuration according to specified key
	 * Creation date: (3/6/2001 4:46:18 PM)
	 * @return com.ibm.pvc.we.ins.und.server.adaptors.transcoder.Transcoder
	 * @param key java.lang.String
	 */
	public static Transcoder createTc(String key) {

		if (Logging._isTracing) {
			Logging._tracer.text(TRANSCODERS,
				"TcFactory","createTc(key)","Entry");
					}

		Transcoder currentTc=loadTc(key);
		return currentTc;
	}
}
```

Everyplace Intelligent Notification Services Transcoder sample code - Generic Transcoder

```
 /*****************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

package com.ibm.pvc.we.ins.und.server.adaptors.transcoder;

import com.ibm.pvc.we.ins.und.server.dispatcher.Msg;
import com.ibm.pvc.we.ins.und.server.adaptors.*;
import com.ibm.pvc.we.ins.GeneralConstants;

import java.util.*;
import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.Transcoder;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import java.io.*;


import com.ibm.pvc.we.ins.und.server.dispatcher.*;

public class GenericTranscoder implements Transcoder, com.ibm.logging.IRecordType
{
        private final static String
                insIBMCopyright=GeneralConstants.insIBMCopyright;

        public static long TRANSCODERS=1<<38;
        protected Properties gwconfig;
        protected String stylesheet;
        protected       Transformer transformer;


        /**
         * GenericTranscoder constructor comment.
         */
        public GenericTranscoder() {
                super();
        }

        /**
         * Insert the method's description here.
         * Format message according to gateway interface
         * For now it just support text format, later rich format
                 * support may be added
         * Creation date: (1/12/2001 2:00:52 PM)
         * @return java.lang.String
         */
        public String format(DeviceAddress addr,Msg msg) {
                 if (Logging._isTracing) {
```

```
                                     Logging._tracer.text(TRANSCODERS,this,"format",
                                                        "Entry");
                                          }
                String message;
                setStylesheet();
                if (stylesheet.equals("NULL"))
                    message=formatMessage(msg);
                else
                    message=formatUseSS(msg);
                if (Logging._isTracing) {
                                     Logging._tracer.text(TRANSCODERS,this,"format",
                                                        "Transformed Message is \n
{0} ",message);
                                          }
                return message;
        }

        /**
         * Insert the method's description here.
         * Default transcoding method, need to be
                 * implemented according to device type
         * @return java.lang.String
         * @param msg com.ibm.pcds.ns.server.src.Msg
         */
        public String formatMessage(Msg msg) {
                return null;
        }

        /**
         * Insert the method's description here.
         * Replace "& and <" with "& and <"
         * @return java.lang.String
         * @param msg com.ibm.pcds.ns.server.src.Msg
         */
        public String restoreEscape(String escapexml) {
                String restorexml=null;

          StringTokenizer st= new StringTokenizer(escapexml,"&");

          while(st.hasMoreTokens())
          {
            String temp= st.nextToken();

            temp=replaceStart(temp,"lt;","<");
            temp=replaceStart(temp,"amp;","&");


            if (restorexml==null)
            {
              restorexml=temp;
            }
            else
            {
              restorexml=restorexml.concat(temp);
            }
          }

          // remove extra char at the end
          return restorexml;
          }

        /**
```

```
         * Insert the method's description here.
         * Replace start delimiter with proper char
         * Eg. replace "&ab" with "&ab"
         * @return java.lang.String
         * @param java.lang.String java.lang.String
         */
        public String replaceStart(String escapexml, String delim,
                String newdelim) {

          String rst=escapexml;

          // replace start delimiter with proper char
          if (escapexml.startsWith(delim))
          {
            rst = new String(newdelim);
            rst=rst.concat(escapexml.substring(delim.length()));
          }

          return rst;
        }

        /**
         * Insert the method's description here.
         * Use stylesheet to do transform & output a string
         * @return java.lang.String
         * @param msg com.ibm.pcds.ns.Msg
         */
        public String formatUseSS(Msg msg) {
                  if (Logging._isTracing) {
                             Logging._tracer.text(TRANSCODERS,this,"formatUseSS",
                                         "Entry");
                                 }


                StreamSource source=new StreamSource(new
                        StringReader(msg.extractML()));
                StreamResult result=new StreamResult(new StringWriter());

                // invoke transformer
                try{
                        transformer.transform(source,result);
                 }
                catch (TransformerException te)
                {
                    //System.out.println(te);
                     if (Logging._isLogging) {
                                     Logging._logger.message(TYPE_ERROR,this,

"formatUseSS","TC_USE_SS_ERROR");
                                         Logging._logger.exception(TRANSCODERS,this,
                                                        "formatUseSS",te);

                                  }
                }
                String msgbody = ((StringWriter)result.getWriter()).toString();

                return msgbody;
        }

        /**
         * Insert the method's description here.
         * @param gwconfig java.util.Properties
         */
        public void setGWConfigFile(Properties gwconfig)
        {
                this.gwconfig = gwconfig;
                }

        /**
         * Insert the method's description here.
         * This method should check gateway configuration
                 * to see if there's a stylesheet tied with
         * this transcoder. Create a transformer for non-null stylesheet
         */
        public void setStylesheet()
        {
                // stylesheet will be set only once during transcoder object life
time
                // if no stylesheet is provided, stylesheet = "NULL"
                if ( stylesheet == null )
                {
                // stylesheet name convention : classname + "_SS"
                String key=this.getClass().getName().concat("_SS");

                // remove package name
                key = key.substring(key.lastIndexOf(".")+1);
                stylesheet = gwconfig.getProperty(key,"NULL");

                //System.out.println("stylesheet is " + stylesheet);

                //debug purpose only
                //System.out.println("Stylesheet key is " +
                        this.getClass().getName().concat("_SS"));
                if (!stylesheet.equals("NULL"))
                    {
                            try
                            {
                            //create transformer
                            TransformerFactory tFactory =
TransformerFactory.newInstance();
                        transformer = tFactory.newTransformer
                                        (new StreamSource(stylesheet));
                            }
                            catch (TransformerConfigurationException tce)
                            {
                                    if (Logging._isTracing) {
                                         Logging._tracer.exception(TRANSCODERS,this,
                                                        "setStylehsheet",tce);
                                         Logging._tracer.stackTrace(TRANSCODERS,this,
                                                        "setStylesheet");
                                  }
                            }

                    }
                }

                }
                 }
```

Everyplace Intelligent Notification Services Transcoder sample code - Wireless Application Protocol

```
 /*****************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors.transcoder;

/**
 * Insert the type's description here.
 * Transcoder will format message specific to communication protocol
 * and adapter implementation
 * of gateway interface.
 */

import java.util.Properties;

import com.ibm.pvc.we.ins.und.server.adaptors.*;
import com.ibm.pvc.we.ins.und.server.adaptors.transcoder.*;

import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import java.io.*;


import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.GeneralConstants;

public class WAPTranscoder extends GenericTranscoder implements Transcoder
{
        private final static String
                insIBMCopyright=GeneralConstants.insIBMCopyright;

        String wapversion;

        /**
         * WAPTranscoder constructor comment.
         */
        public WAPTranscoder()
        {
                super();

        }

        /**
         * Insert the method's description here.
         * Set WAP version
         */
        public String addVersion(String message,String ver)
        {
                        // if no version is provided or not a valid version,
                                // WAP1.1 is assumed

                        if ( ver == null )
                            ver = "1V1";
                        else if ( !ver.equals("1V2"))
                            ver = "1V1";


                    // insert version
                StringBuffer tempmsg= new StringBuffer(message);
                int offset = message.indexOf("<wml>");
                if (ver.equals("1V1"))
                {
                tempmsg.insert(offset,
                        "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML" +
                        "1.1//EN\"\n\"http://www.wapforum.org/DTD/wml_1.1.xml\">\n");
                }
                else if (ver.equals("1V2"))
                {
                tempmsg.insert(offset,
                        "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML" +
                        "1.2//EN\"\n\"http://www.wapforum.org/DTD/wml12.dtd\">\n");
                }

                // Debug purpose
                return tempmsg.toString();


           }

        /**
         * Insert the method's description here.
         * Format message according to gateway interface
         * For now it just support text format, later rich format
                 * support may be added
         * @return java.lang.String
         */
        public String format(DeviceAddress addr,Msg msg) {
                String message;

                 if (Logging._isTracing) {
                                    Logging._tracer.text(TRANSCODERS,
                                                       this,"format","Entry");
                                         }
                // check if user provides a style sheet
                setStylesheet();
                if (stylesheet.equals("NULL"))
                    message=formatMessage(msg);
                else
                    message=formatUseSS(msg);

                message = addVersion(message,addr.getVersion());
                 if (Logging._isTracing) {
                        Logging._tracer.text(TRANSCODERS,
                                this,"format","Transformed Message is \n {0}
",message);
                                         }
                return message;
        }
        /**
         * Insert the method's description here.
         * @return java.lang.String
         * @param msg com.ibm.pcds.ns.Msg
         */
        public String formatMessage(Msg msg) {

                 if (Logging._isTracing) {
                                   Logging._tracer.text(TRANSCODERS,

this,"formatMessage","Entry");
                                          }
                String notificationML;

                 notificationML=msg.extractML();


                // WML deck has been tested with Nokia toolkit 2.0
                StringBuffer b=new StringBuffer();
                b.append("<?xml version=\"1.0\"?>\n");

                b.append("<wml>\n<card id=\"Notification\"
                        title=\"Message\">\n");
                b.append("<p>\n");


                b.append("From: ").append(msg.fromUserid).append("<br/>\n");
                b.append("Subject: ").append(msg.subject).append("<br/>\n");

                String
                        msgbody=notificationML.substring(notificationML.indexOf
                        ("<text>")+6,notificationML.indexOf
                        ("</text>"));
                b.append("Msg: ").append(msgbody).append("<br/>\n");

                // allow multiple urls

                int index=notificationML.indexOf("<url>");
                while ( index != -1 )
                {
                String url=notificationML.substring(index+5,notificationML.indexOf
                ("</url>",index));
                b.append("\t").append(url).append("<br/>\n");
                index=notificationML.indexOf("<url>",index+1);
                }

                b.append("</p>\n").append("</card>\n</wml>\n");
          return b.toString();
        }
}
```

Everyplace Intelligent Notification Services Transcoder sample code - Instant Messaging / Sametime

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors.transcoder;

import com.ibm.pvc.we.ins.und.server.dispatcher.Msg;
import com.ibm.pvc.we.ins.und.server.dispatcher.Logging;
import com.ibm.pvc.we.ins.GeneralConstants;

/**
 * Insert the type's description here.
 * Transcoder for Instant Messaging Sametime.
 */
public class IMSTTranscoder extends GenericTranscoder
{
        private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;

      /**
       * IMSTTranscoder constructor comment.
       */
      public IMSTTranscoder() {
              super();
      }

      public String formatMessage(Msg msg)
      {
                if (Logging._isTracing) {
                         Logging._tracer.text(TRANSCODERS,this,"formatMessage",
                                         "Entry");
                                  }
        String notificationML=msg.extractML();

        StringBuffer b=new StringBuffer();


        b.append("\tFrom: ").append(restoreEscape(msg.fromUserid)).append("\n");
        b.append("\tSubject: ").append(restoreEscape(msg.subject)).append("\n");

        String
                msgbody=notificationML.substring(notificationML.indexOf("<text>")+6,
notificationML.indexOf("</text>"));
        msgbody=restoreEscape(msgbody);
        b.append("\tMsg: ").append(msgbody).append("\n");
        // allow multiple urls

        int index=notificationML.indexOf("<url>");
        while ( index != -1 )
        {
        String
                url=notificationML.substring(index+5,notificationML.indexOf("</url>",
        index));
        url=restoreEscape(url);
        b.append("\t").append(url).append("\n");
        index=notificationML.indexOf("<url>",index+1);
        }

        return b.toString();

        }
}
```

Everyplace Intelligent Notification Services Transcoder sample code - Short Messaging Service

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors.transcoder;

import com.ibm.pvc.we.ins.und.server.dispatcher.Msg;
import com.ibm.pvc.we.ins.und.server.dispatcher.DeviceAddress;
import com.ibm.pvc.we.ins.und.server.dispatcher.Logging;
import com.ibm.pvc.we.ins.GeneralConstants;

/**
 * Insert the type's description here.
 * Transcoder for SMS
 */
public class SMSTranscoder extends GenericTranscoder
{
   private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;

        /**
         * SMSTranscoder constructor comment.
         */
        public SMSTranscoder() {
                super();
        }
        /**
         * Insert the method's description here.
         * Modify text message to fit SMS device requirements.
         * If the message is just too long to fit, return a string starting with"!".
         */
        public String fitSMSLength(String message)
        {

                // if message too long, try to shorten message by excluding subject
                if ( message.length()>160 ) {
                        message=message.substring(0,message.indexOf("Sub:")-1).concat
                        (message.substring(message.indexOf("Msg:")));
                        }

                // if message still too long: inappropriate for SMS delivery
                if ( message.length()>160 ) {
                                message=new String("!MESSAGE TOO LONG FOR SMS
DEVICE");
                                        }
        return message;

     }
  /**
   * Insert the method's description here.
   * Format message according to gateway interface
   * For now it just support text format, later rich
         * format support may be added
   * @return java.lang.String
   */
  public String format(DeviceAddress addr,Msg msg) {
        String message;
        if (Logging._isTracing) {
                         Logging._tracer.text(TRANSCODERS,this,
                                       "format","Entry");
                              }
        // check if user provides a style sheet
        setStylesheet();
        if (stylesheet.equals("NULL"))
            message=formatMessage(msg);
        else
            message=formatUseSS(msg);

        message = fitSMSLength(message);



         if (Logging._isTracing) {
                Logging._tracer.text(TRANSCODERS,this,
                        "format","Transformed Message is \n {0} ",message);
                              }

        return message;
  }
  /**
   * Insert the method's description here.
   * Creation date: (2/9/2001 1:14:51 PM)
   */
  public String formatMessage(Msg msg)
  {
        if (Logging._isTracing) {
                 Logging._tracer.text(TRANSCODERS,this,
                        "formatMessage","Entry");
                              }

        String notificationML=msg.extractML();
        // create message from Msg
        StringBuffer b=new StringBuffer();
        b.append("From: ").append(restoreEscape(msg.fromUserid)).append(
'\n');
        b.append("Subject: ").append( restoreEscape(msg.subject)).append(
'\n');
        b.append("Msg: ").append( restoreEscape(msg.umsg)).append('\n');

        // allow multiple urls
        int index=notificationML.indexOf("<url>");
        while ( index != -1 )
        {
        String url=notificationML.substring(index+5,notificationML.indexOf
        ("</url>",index));
  url=restoreEscape(url);
        b.append("\t").append(url).append("\n");
```

```
                index=notificationML.indexOf("<url>",index+1);
                }

                return b.toString();

                }
}
```

Everyplace Intelligent Notification Services Transcoder sample code - SMTP Email

```
 /*****************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/

package com.ibm.pvc.we.ins.und.server.adaptors.transcoder;

/**
 * Transcode message according to Mail Bean API.
 */

import com.ibm.pvc.we.ins.und.server.dispatcher.Msg;
import com.ibm.pvc.we.ins.und.server.dispatcher.DeviceAddress;
import com.ibm.pvc.we.ins.und.server.dispatcher.Logging;
import com.ibm.pvc.we.ins.GeneralConstants;

public class EMAILTranscoder extends GenericTranscoder
{
        private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;

        /**
         * EMAILTranscoder constructor comment.
         */
        public EMAILTranscoder() {
                super();
        }
        /**
         * Insert the method's description here.
         * Returns a message body which includes message text and urls
         */
        public String formatMessage(Msg msg)
        {
                if (Logging._isTracing) {
                   Logging._tracer.text(TRANSCODERS,this,
                                        "formatMessage","Entry");
                                          }

                String notificationML=msg.extractML();

                StringBuffer b=new StringBuffer();

                String msgbody=notificationML.substring(notificationML.indexOf
                ("<text>")+6,notificationML.indexOf("</text>"));
          msgbody=restoreEscape(msgbody);
                b.append("Msg: ").append(msgbody).append("\n");

                // allow multiple urls
                int index=notificationML.indexOf("<url>");
                while ( index != -1 )
                {
                String url=notificationML.substring(index+5,notificationML.indexOf
                        ("</url>",index));
          url=restoreEscape(url);
                b.append("\t").append(url).append("\n");
                index=notificationML.indexOf("<url>",index+1);
                }

                b.append("PLEASE DO NOT REPLY TO THIS NOTE.\n");

                return b.toString();
                }
}
```

Everyplace Intelligent Notification Services Transcoder sample code - Email Push

```
/******************************************************************************
 *  IBM WebSphere Everyplace Server Version 2.1 (PID 5724-B07)                *
 *  (c) Copyright IBM Corporation 2000, 2001.  All Rights Reserved            *
 *                                                                            *
 *  Licensed Materials - Property of IBM                                      *
 *                                                                            *
 *  US Government Users Restricted Rights - Use, duplication or disclosure    *
 *  restricted by GSA ADP Schedule Contract with IBM Corporation.             *
 ******************************************************************************
*/
package com.ibm.pvc.we.ins.und.server.adaptors.transcoder;

import com.ibm.pvc.we.ins.und.server.dispatcher.*;
import com.ibm.pvc.we.ins.GeneralConstants;

/**
 * Insert the type's description here.
 * Transcoder for EMail.
 */
public class MailPushTranscoder extends GenericTranscoder
{
        private final static String insIBMCopyright=GeneralConstants.insIBMCopyright;

        /**
         * MailPushTranscoder constructor comment.
         */
        public MailPushTranscoder() {
                super();
        }

        /**
         * Insert the method's description here.
         * Default transcoding method, need to be implemented according to device
type
         * @return java.lang.String
         * @param msg com.ibm.pcds.ns.server.src.Msg
         */
        public String formatMessage(Msg msg) {

                if (Logging._isTracing) {
                    Logging._tracer.text(TRANSCODERS,this,
                                        "formatMessage","Entry");
                                          }

                String notificationML=msg.extractML();

                // create message from Msg
                StringBuffer b=new StringBuffer();

                b.append("Msg: ").append( restoreEscape(msg.umsg)).append("\n");

                // allow multiple urls
                int index=notificationML.indexOf("<url>");
                while ( index != -1 )
                {
                String
                url=notificationML.substring(index+5,notificationML.indexOf("</url>",
        index));
          url=restoreEscape(url);
                b.append("\t").append(url).append("\n");
                index=notificationML.indexOf("<url>",index+1);
                }
                b.append("PLEASE DO NOT REPLY TO THIS NOTE.\n");
                return b.toString();
        }
}
```

WAP transcoder stylesheet

The stylesheet file, *WAPTranscoder_SS.xsl* contains the following stylesheet code.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:output method="xml" indent="yes" />
<xsl:template match="message">
<wml>
   <card id="Notification" title="Message">
     <p>
     From:<xsl:value-of select="from"/>
     <br/>
     Sub: <xsl:value-of select="subject"/>
     <br/>
     Msg: <xsl:value-of select="text"/>
     <br/>
     <xsl:for-each select="url">
     <xsl:text>
     </xsl:text><xsl:value-of select="."/>
        <br/>
     </xsl:for-each>
     </p>
   </card>
</wml>
</xsl:template>

</xsl:stylesheet>
```

SMS transcoder stylesheet

The stylesheet file, *SMSTranscoder_SS.xsl* contains the following stylesheet code.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:output omit-xml-declaration="yes" method="text"/>
<xsl:template match="message">
From: <xsl:value-of select="from"/>
Sub: <xsl:value-of select="subject"/>
Msg: <xsl:value-of select="text"/>
<xsl:for-each select="url">
      <xsl:text>
        </xsl:text><xsl:value-of select="."/>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

E-MAIL transcoder stylesheet

The stylesheet file, *EMAILTranscoder_SS.xsl* contains the following stylesheet code.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:output omit-xml-declaration="yes" method="text"/>
<xsl:template match="message">
Msg: <xsl:value-of select="text"/>
      <xsl:for-each select="url">
      <xsl:text>
        </xsl:text><xsl:value-of select="."/>
      </xsl:for-each>
<xsl:text>
PLEASE DO NOT REPLY TO THIS NOTE
</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

IMS transcoder stylesheet

The stylesheet file, *IMSTranscoder_SS.xsl* contains the following stylesheet code.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:output omit-xml-declaration="yes" method="text"/>
<xsl:template match="message">
        From: <xsl:value-of select="from"/>
        Sub: <xsl:value-of select="subject"/>
        Msg: <xsl:value-of select="text" />
     <xsl:for-each select="url">
     <xsl:text>
        </xsl:text><xsl:value-of select="." />
     </xsl:for-each>
<xsl:text>

</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

```
// Intelligent Notification Service API
import com.ibm.pvc.we.ins.*;
import java.io.*;

/**
 * Basic sample demonstrating use of the WES Intelligent Notification Service client
API.
 */
public class Everyplace Intelligent Notification ServicesTest {

        public static void main(String[] args){
                // create a client service stub for the specified UND server and port
                NotificationService ns = new
NotificationService("res.raleigh.ibm.com",12049);

                // specify delivery options
                DeliveryOptions opts = new DeliveryOptions();

                // select devices to which message is delivered
                opts.devices = opts.IM+opt.WAP;

                // select message priority
                opts.priority = opts.NORMAL;

                // opts.ANY indicates that the server should send the message to any
one device
                // specified in opts.devices.The server will loop through the devices
and try to
                // send the message, and will stop once the message is successfully
delivered.
                //
                // specify opts.ALL to send message to all specified devices.
                opts.multiDevices = opts.ANY;

                // create message in NotificationML format
                String message = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>"
                                           + "<message>"
                                           + "<to>ryan</to>"
                                           + "<from>jpboone</from>"
                                           + "<subject>meeting cancelled</subject>"
                                           + "<text> I'm going
home</text></message>";
                try {
                        // send the message with the specified delivery options
                        ns.sendMessage(message,opts);
                }
                // an IOException will be thrown if the NotificationService has
                // trouble communicating with the server
                catch (IOException e)
                {
                        e.printStackTrace();
                }

        }


}
```