Mobile RSS Reader Feed Content UI 1.0 Component Specification

1. Design

The RSS Feed Content UI component will provide the user interface primarily for a user to carry out CRUD operations that can be performed on feeds as well as manage getting and setting the data attributes associated with a feed unto the data model.

This design provides the following features:

- Display feed and feed entries for a specific subscribed feed. This can be done by setting and then displaying a ViewFeedList instance.
- Feed entries are displayed in a scrollable list categorized by the feed which is the container for the feed entries. This is can be done using the title of the screen (i.e. TagName – FeedName).
- Feed entries are sortable. ViewFeedList provides the ability to sort them either by name or by date.
- Users will be allowed to intentionally mark feed entries as read or unread as well as toggle between these settings for all feeds. It is responsibility of the controller to update the Screen when this happens.
- The ReplyFeedEntryForm provides a Screen to post response to a feed content entry.
- The RSS Feed Content UI component display user-friendly and simple error messages to the user(using Jadabs-Log4j-J2ME). It is responsibility of the controller to provide and display the appropriate error Screen.
- The Mobile RSS Reader Controller component is responsible for navigation between screens and it will run as a Midlet while this component and other UI components will be subclasses of the screen component.
- When the controller has to process something that blocks the application, it should display an indication to the user that the action is on-going.
- Controller managed data structures that are potentially accessed by multiple concurrent thread needs to be thread safe, that is synchronized to avoid race-conditions or dead-locks.
- The controller must be thread safe as it will be responsible for user interaction workflow as well as the scheduled retrieval of feed updates at specified polling intervals.

Note to developers:

- Mobile devices can be subject to low resolution displays, low bandwidth connections and slow processing. These factors need to be taken into consideration.
- The lowest resolution that will be supported by this application is 176 x 208. Resolutions as high as 320 x 320 are also expected.

1.1 Design Patterns

This component implements the **View** in the **Model-View-Controller** pattern; the **Model** is represented by Mobile RSS Feed Content component while Mobile RSS Reader

Controller component is the **Controller**.

The **Observer** pattern is used to allow the Controller have full control on screens flow and model updating. This is achieved by registering at least a MIDP CommandListener to each Screen

1.2 Industry Standards

XML, RSS 2.0, MIDP 2.0, CLDC 1.1

1.3 Required Algorithms

1.3.1View Feed Screen

See wireframe 2.4.2 for detailed page content and flow.

Creating the screen:

- Call super(title,Choice.IMPLICIT).
- Add the “View Feed” command of type BACK to the List.
- Add the “Toggle All” command of type SCREEN to the List.
- Add the “Refresh” command of type SCREEN to the List.
- Add the “Sort By Name” command of type SCREEN to the List.
- Add the “Sort By Date” command of type SCREEN to the List.
- Add the “Back” command of type SCREEN to the List.
- Add the “Help” command of type HELP to the List.
- Call setFeed().

Setting the feed:

- Set this.feedEntryId to RSSFeedContentEntry.getObjectId()
- Set the title of the List.
- Set an empty FilterListModel.
- Get all RSSFeedContentEntrys from RSSFeedContent parameter.
- For each RSSFeedContentEntry, create a FilterListEntry(new String[]{RSSFeedContentEntry.getName(), RSSFeedContentEntry.getUpdatedDate(), RSSFeedContentEntry.getObjectId(), RSSFeedContentEntry.isRead.toString}) and add it to model class field.

Updating UI:

- Delete all elements from the List.
- For each FilterListEntry of the model: append a new String(entry.getColumn(0) + “ - “ + entry.getColumn(1)); if entry.getColumn(3) is "false" set it selected making its Font to be bold.

1.3.2Read Feed Entry Screen

See wireframe 2.4.3 for detailed page content and flow.

Creating the screen:

- Call super(title).
- Add the “Refresh” command of type BACK to the Form.

- Add the “Mark Read/Unread” command of type ITEM to the Form.
- Add the “Next” command of type SCREEN to the Form.
- Add the “Previous” command of type SCREEN to the Form.
- Add the “Reply” command of type SCREEN to the Form.
- Add the “Back” command of type SCREEN to the Form.
- Add the “Help” command of type HELP to the Form.
- Call setFeedEntry().

Setting the feedEntry:

- Clear the form
- Set the title of the Form.
- Append headerImage.
- Set this.feedEntryId to RSSFeedContentEntry.getObjectId()
- Append a new String RSSFeedContentEntry.getName() + “ - “ + RSSFeedContentEntry.getUpdatedDate() + “\n” .
- Append a new String RSSFeedContentEntry.getDescription() .

1.3.3Reply to Feed Entry Screen

See wireframe 2.4.4 for detailed page content and flow.

Creating the screen:

- Call super(title).
- Add the “Post” command of type OK to the Form.
- Add the “Clear” command of type BACK to the Form.
- Call setFeedEntry().

Setting the feedEntry:

- Clear the form
- Set the title of the Form.
- Append headerImage.
- Append a new String RSSFeedContentEntry.getName() + “ - “ + RSSFeedContentEntry.getUpdatedDate() + “\n”.
- Append a new TextField("Comments:",null, maxSize,Item.LAYOUT_DEFAULT).
- Append a new String RSSFeedContentEntry.getDescription().

1.3.4Command construction

Commands are displayed as soft buttons, and every mobile device has its own way to display them. In order for the menu to be as consistent as possible between different devices, developers are encouraged to assign the same type to all commands (ITEM for instance) and to use the priority argument when constructing them, assigning a different priority to each one.

1.3.5Logging

Logging is performed using Jadabs-Log4j-J2ME

Logging is performed when a method is entered or exited at level INFO or when an exception is caught at level ERROR, assuming all the logger configuration and initialization is done by Mobile RSS Reader Controller, that should configure a logger for each Screen class of this component with the related DEFAULT_LOG_NAME.

To obtain the Logger in case the related static field is null, simply call Logger.getLogger(this.DEFAULT_LOG_NAME).

For more information on logging mechanism please visit:
http://jadabs.berlios.de/jadabs-cldc/multiproject/log4j-j2me/index.html

1.4 Component Class Overview

ViewFeedList:

ViewFeedList is a List Screen that extends FilterList. It merely displays feed and feed entries, allowing sorting by date or by name. The feed to be viewed is setted at creation time, but can always be changed by setter. When the user takes an action the command is passed to the registered listener of type CommandListener. Every time there is a call to the listener, an Alert should be shown to user by the listener to let it know action is on-going or in case an exception is caught.

It provides methods to get the selected feed entry Id in case the listeners or the controller need it for processing (for instance after a selection event) or the feed id. If an entry is unread it is shown in bold Font, else with the default Font. It is responsibility of the controller to set an entry as read or as unread if the user select the related command on it: for this setSelectedFeedEntryStatus is provided. At opposite, if setFeed() is called with the changed model as argument, bold Font is automatically set on unread entries.

If "Toggle All" is fired, the controller should update the model switching the isRead field and then call toggleAll() to correctly display the entries.

When the "Sort By Date" or "Sort By Name" command event is fired the related method should be called on this screen in order to sort the displayed List.

When "View Feed" is fired the ReadFeedEntryForm should be set as the current Displayable.

When "Refresh" is fired the controller should update the feed and set it in this Screen.

When "Back" is fired the controller should display the previous Screen.

When "Help" is fired, the controller should display a custom Screen explaining the mean of the commands.

The Mobile Filter List component provides an abstract method to be implemented (updateUI()). Here the list to be displayed is set, based on the ordered model. It is called after ordering by superclass.

Logging is performed when a method is entered or an exception is caught, assuming all the logger configuration and initialization is done by Mobile RSS Reader Controller. See details in 3.1.5 of CS.

This class is mutable, so it is not thread-safe.

ReadFeedEntryForm:

ReadFeedEntryForm is a Screen that extends Form. It merely displays the content of a feed entry. The feed entry to be viewed is setted at creation time, but can always be changed by setter. When the user takes an action the command is passed to the registered listener of type CommandListener. Every time there is a call to the listener, an Alert should be shown to user by the listener to let it know action is on-going or in case an exception is caught.

It provides a method to get the ID of the diplayed entry in case the listeners or the controller need it for processing.

When "Refresh" is fired the controller should update the entry and set it in this Screen.

When “Mark Read/Unread” is fired the controller should update the model.

When “Next” or "Previous" is fired the controller should display the next/previous entry.

When "Reply" is fired the controller should display the appropriate ReplyFeedEntryForm.

When "Back" is fired the controller should display the previous Screen.

When "Help" is fired, the controller should display a custom Screen explaining the mean of the commands.

Logging is performed when a method is entered or an exception is caught, assuming all the logger configuration and initialization is done by Mobile RSS Reader Controller. See details in 3.1.5 of CS.

This class is mutable, so it is not thread-safe.

ReplyFeedEntryForm:
ReplyFeedEntryForm is a Screen that extends Form. It displays input text field for the reply followed by the content of the feed entry. The feed entry to be replied to is setted at creation time, but can always be changed by setter. When the user takes an action the command is passed to the registered listener of type CommandListener. Every time there is a call to the listener, an Alert should be shown to user by the listener to let it know action is on-going or in case an exception is caught.

It provides a method to get the ID of the diplayed entry and one method to get the message to post, in case the listeners or the controller need them for processing.

When "Post" is fired, the controller should post the message to the feed.

When "Clear" is fired, the controller should call clear() in order to clear the displayed text field.

Logging is performed when a method is entered or an exception is caught, assuming all the logger configuration and initialization is done by Mobile RSS Reader Controller.

This class is mutable, so it is not thread-safe.

1.5 Component Exception Definitions

None.

1.6 Thread Safety

This component is not thread safe because the provided Screens are mutable. It is a responsibility of the controller component to use it in a thread-safe manner; a possible way to do this is to synchronize on a locking object for each Screen when accessing it by any of its methods.

2. Environment Requirements

2.1 Environment

- Development language: J2ME 2.1, MIDP 2
- Compile target: MIDP 2.0 and CLDC 1.1
- Runtime environment: Any J2ME MIDP 2.0 CLDC 1.1 compatible device

2.2 TopCoder Software Components

- Mobile RSS Reader Controller Component version 1.0: this is an indirect dependency, as this component controls the screens flow.
- Mobile RSS Feed Content Component version 1.0: this is an indirect dependency, as it is used by Mobile RSS Reader Controller component.
- Mobile HTTP Handler Component version 1.0: this is an indirect dependency, as it is used by Mobile RSS Feed Content component.
- Mobile Data Broker: defines RSSFeedContent and RSSFeedContentEntry entities.

NOTE: The default location for TopCoder Software component jars is../lib/tcs/COMPONENT_NAME/COMPONENT_VERSION relative to the component installation. Setting the tcs_libdir property in topcoder_global.properties will overwrite this default location.

2.3 Third Party Components

- J2ME – MIDP LCDUI package: used for creating and showing UI.
- Jadabs-Log4j-J2ME: used for logging.

NOTE: The default location for 3rd party packages is ../lib relative to this component installation. Setting the ext_libdir property in topcoder_global.properties will overwrite this default location.

3. Installation and Configuration

3.1 Package Name

com.topcoder.mobilerssreader.rssfeedcontentui

3.2 Configuration Parameters

None.

3.3 Dependencies Configuration

None.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Execute ‘ant test’ within the directory that the distribution was extracted to.

4.2 Required steps to use the component

Follow demo.

4.3 Demo

```
/**
 * <p>
 * Default constructor.
 * </p>
 */
```

```
public DemoTest() {
    OSGiContainer osgicontainer = OSGiContainer.Instance();
    osgicontainer.setProperty("log4j.priority", "INFO");
    osgicontainer.startBundle(new LogActivator());
    headerImage = TestHepler.getImage("/topcoder.gif");
}

/**
 * <p>
 * This method should be called when, from an RSSSubcription Screen, the user
wants to view the list of entries of a
 * specific feed.
 * </p>
 *
 * @param title
 *          title to construct the list page.
 * @param feed
 *          feed to construct the list page.
 */
public void showFeedList(String title, RSSFeedContent feed) {
    viewFeed = new ViewFeedList(title, feed);
    viewFeed.addCommand(new Command("Log", Command.OK, 0));
    viewFeed.setCommandListener(new ViewFeedListener(this));
    setCurrent(viewFeed);
}

/**
 * <p>
 * This method should be called when, from the list Screen, the user wants
to view the selected entry.
 * </p>
 *
 * @param title
 *          title to construct the read page.
 * @param entry
 *          entry to construct the read page.
 */
public void showFeedEntry(String title, RSSFeedContentEntry entry) {
    readEntry = new ReadFeedEntryForm(title, entry, headerImage);
    readEntry.addCommand(new Command("Log", Command.OK, 0));
    readEntry.setCommandListener(new ReadEntryListener(this));
    setCurrent(readEntry);
}

/**
 * <p>
 * This method should be called when, from the read Screen, the user wants
to reply the read entry.
 * </p>
 *
 * @param title
 *          title to construct the reply page.
 * @param entry
 *          entry to construct the reply page.
 */
public void replyFeedEntry(String title, RSSFeedContentEntry entry) {
    replyEntry = new ReplyFeedEntryForm(title, entry, headerImage, 100);
```

```
    replyEntry.addCommand(new Command("Log", Command.OK, 0));
    replyEntry.setCommandListener(new ReplyEntryListener(this));
    setCurrent(replyEntry);
}

/**
 * <p>
 * set the display to current display.
 * </p>
 *
 * @param display
 *          the display to set to current.
 */
public void setCurrent(Displayable display) {
    Display.getDisplay(this).setCurrent(display);
}

/**
 * <p>
 * Show the log canvas.
 * </p>
 */
public void viewLog() {
    setCurrent(Logger.getLogCanvas());
}

/**
 * <p>
 * Start up of the application.
 * </p>
 *
 * @throws MIDletStateChangeException
 *           if any error occurs.
 */
protected void startApp() throws MIDletStateChangeException {
    String title = "all the feed content entries";
    entries = new RSSFeedContentEntry[10];
    boolean read = true;
    for (int i = 0; i < entries.length; i++) {
        entries[i] = new RSSFeedContentEntry("name:" + i, "objectId:" + i,
"description:" + i, new URL(
                "http://www.topcoder.com/"), read, new Date());
        read = !read;
    }
    RSSFeedContent feed = new RSSFeedContent("name", "objectId",
"description",
            new URL("http://www.topcoder.com/"), new Date(), entries);
    showFeedList(title, feed);
}

/**
 * <p>
 * CommandListener for the list page.
 * </p>
 *
 * @author TCSDEVELOPER
 * @version 1.0
```

```
 */
private class ViewFeedListener implements CommandListener {
    /**
     * <p>
     * the MIDlet instance.
     * </p>
     */
    private DemoTest controller;

    /**
     * Constructor of the listener.
     *
     * @param controller
     *          the MIDlet instance.
     */
    public ViewFeedListener(DemoTest controller) {
        // initialize something if you need
        this.controller = controller;
    }

    /**
     * <p>
     * The command action method of the listener.
     * </p>
     */
    public void commandAction(Command command, Displayable displayable) {
        // For instance, if the command is "View Feed"
        if (command.getLabel().equals("View Feed")) {
            currentSelectedIndex = viewFeed.getSelectedIndex();
            RSSFeedContentEntry entry = entries[currentSelectedIndex];
            // Let the controller set and display the new Screen
            controller.showFeedEntry(entry.getName(), entry);
        } else if (command.getLabel().equals("Log")) {
            controller.viewLog();
        }
    }
}

/**
 * <p>
 * CommandListener for the read page.
 * </p>
 *
 * @author TCSDEVELOPER
 * @version 1.0
 */
private class ReadEntryListener implements CommandListener {
    /**
     * <p>
     * the MIDlet instance.
     * </p>
     */
    private DemoTest controller;

    /**
     * <p>
     * The constructor.
```

```
     * </p>
     */
    public ReadEntryListener(DemoTest controller) {
        this.controller = controller;
    }

    /**
     * <p>
     * The command action method of the listener.
     * </p>
     */
    public void commandAction(Command command, Displayable displayable) {
        // For instance, if the command is "Reply"
        if (command.getLabel().equals("Reply")) {
            RSSFeedContentEntry entry = entries[currentSelectedIndex];
            // Let the controller set and display the new Screen
            controller.replyFeedEntry(entry.getName(), entry);
        } else if (command.getLabel().equals("Back")) {
            controller.setCurrent(viewFeed);
        } else if (command.getLabel().equals("Log")) {
            controller.viewLog();
        }
    }
}

/**
 * <p>
 * CommandListener for the reply page.
 * </p>
 *
 * @author TCSDEVELOPER
 * @version 1.0
 */
private class ReplyEntryListener implements CommandListener {
    /**
     * <p>
     * the MIDlet instance.
     * </p>
     */
    private DemoTest controller;

    /**
     * <p>
     * Default constructor.
     * </p>
     */
    public ReplyEntryListener(DemoTest controller) {
        this.controller = controller;
    }

    /**
     * <p>
     * The command action method of the listener.
     * </p>
     */
    public void commandAction(Command command, Displayable displayable) {
        // For instance, if the command is "Back", return to
```

```
        if (command.getLabel().equals("Clear")) {
            replyEntry.clear();
        } else if (command.getLabel().equals("Post")) {
            replyEntry.append("\nMessage:" + replyEntry.getPostMessage() + "
Posted !");
        } else if (command.getLabel().equals("Log")) {
            controller.viewLog();
        }
    }
}
```

5. Future Enhancements

Any future update follows from requirements changes to Mobile RSS Reader application.