The Bundle activity: Requirement Specification

1. Introduction

This document describes the behavior of a new activity called the Bundle activity for the XO Laptop, the low-cost, low-power laptop developed by the One Laptop per Child (OLPC) association. The basic premise of this activity is to enable users to decompress a selection of objects from an archive for insertion into the Journal, and to place a collection of objects from the Journal into a compressed bundle of their own. The Bundle activity will prove quite useful for sharing; instead of sharing multiple objects individually, one can share objects as a single collection.

2. Environment characteristics

2.1. Hardware

The XO hardware is detailed on the hardware section of the OLPC Wiki (http://wiki.laptop.org/go/Hardware_specification), but the feature that this activity must heed the most is the 256 MiB dynamic RAM.

2.2. Operating System

The XO uses a recent version (2.6.xx) of Linux and a Fedora base environment.

2.3. Compression and archiving libraries

We plan to use an open source compression and archiving library to compress, decompress, archive, and extract objects for this activity.

2.4. User expertise

The OLPC human interface guidelines (http://wiki.laptop.org/go/OLPC_Human_Interface_Guidelines) stress that developers must understand their audience and suggest that the typical user is inexperienced, young, and international. Hence, the Bundle activity certainly should obey the fundamental principles behind the Sugar interface as well—use should be discoverable and intuitive, scalable across ages from five or six to mid-teens, and visuals should rely heavily on pictographic icons and images that transcend language and culture.

3. Input/Output and System Modes

3.1. User input

This activity is poised to be completely mouse-driven. Along with clicking the buttons featured by the activity, user input is just selection—the user will either select objects to erase or extract from a bundle or select objects to add to a bundle he or she created. Selection is handled by the point-and-click capabilities of the mouse.

3.2. Output

3.2.1. User interface
Output is experienced through the display of the Bundle activity and new entries appearing in the Journal. The client provided no specifics or mockups for what the Bundle activity should look like, but the design surely should follow the Sugar interface philosophy and conventions. We provide a mockup at the end of this document.

3.2.2. Extraction
When the user extracts objects from a bundle, the objects will appear as entries in the Journal.

3.2.3. Creation
When the user creates a bundle, the bundle will appear as a single instance of the Bundle activity in the Journal, and the objects added to the bundle will remain as distinct entries in the Journal.

4. Software functions

4.1. Importing a bundle
This is actually Journal behavior, but worthy of mention for our purposes. The user can import a bundle from the Web, a USB drive, or from a friend who is sharing a bundle. When he or she does this, the bundle appears as the most recent entry in the Journal as an instance of the Bundle activity, stamped with the time at which it was imported.

4.2. Creating a new bundle

4.2.1. Starting a new Bundle activity
There should be an icon in the actions portion of the Frame that the user can click to begin a new Bundle activity. Starting the Bundle activity in this way puts it in “create” mode, and the bundle should be initially empty.

4.2.2. Usable functionality
The “create” mode means that adding, erasing, packing, and changing archive format are allowed but extracting is disallowed. Hence, the Extract and Extract All buttons should be inoperative and “grayed out.” Adding, erasing, packing, and choosing an archive format are detailed in section 4.3., and all behave identically in both “create” and “edit” mode.

4.2.2.1. Empty bundle
If the bundle is empty, all buttons except Add are not meaningful and should be grayed out. This effect improves utility since it clearly shows the user what actions are allowed, and it is more graceful then handling each button as a separate, unexpected case if it pressed when the bundle is empty.

4.2.3. Finalizing the bundle

When the user is finished making the bundle, he or she can conclude the Bundle activity with either the Pack button or the Stop activity button.

4.2.3.1. Pack button

Pressing the Pack button compresses the included objects and archives them in the chosen format. The standard circle of dots should appear while the user waits for the compression to finish; when it does, a dialog box (perhaps just a pictogram, not words) should indicate the result.

4.2.3.1.1. Success

If the compression is successful, the dialog box should display a large green check mark and time out after two or three seconds. Then, the Bundle activity is exited, and the user returns to the standard view of the Journal, which will now feature this bundle as the most recent entry. Like all other activities, this instance of the Bundle activity should be stamped with the time at which the activity ended.

4.2.3.1.2. Failure

If the compression fails, the dialog box should display a large red X and time out after two or three seconds. Then, the user just returns to the Bundle activity, which looks exactly as it did before the Pack button was pressed.

4.2.3.2. Stop activity button

Pressing the Stop activity button *silently* packs the bundle and exits identically to exiting after a successful pack.

4.3. Opening an existing bundle

4.3.1. Starting the Bundle activity

The user can open and edit an existing bundle in the Journal, which puts the Bundle activity in “edit” mode. Note that there is no difference between an imported bundle and a self-created bundle here. The bundle contents should be displayed.

4.3.2. Usable functionality

In “edit” mode, all buttons and the drop-down menu are operative.

4.3.2.1. Empty bundle

As in “create” mode, an empty bundle is a special case because all buttons except Add become inoperative and grayed out.

4.3.3. Add

Clicking the Add button brings up the Object Chooser, which looks a lot like the Journal.

4.3.3.1. Choosing an object

The user can click an object to add it to the bundle. Once an object is clicked, the Bundle activity window returns, and the object is added to the list of objects in the bundle.

4.3.3.2. Closing the Object Chooser

If the user decides not to add an object to the bundle, he or she can close the Object Chooser. The Bundle activity window returns, and the contents of the bundle should be unaltered.

4.3.4. Erase

The user can click an object in the bundle, followed by the Erase button to remove the selected object from the bundle. The object will be deleted and will disappear from the bundle.

4.3.5. Extraction

The user can decompress and extract objects from the bundle using the Extract and Extract All buttons. The objects will still exist in the bundle and will appear in the Journal as distinct entries, too. As Journal entries, they will be stamped with the time at which they were last modified, not when they were extracted.

4.3.5.1. Extract

The user can click an object in the bundle, followed by the Extract button to decompress and extract only the selected object from the bundle.

4.3.5.2. Extract All

Clicking the Extract All button decompresses and extracts *all* objects in the bundle.

4.3.6. Selecting archive format

The user can use a drop-down menu to select an archive format to use for the bundle. Possible formats include .zip, .tar, and .gz, and .zip should be the default option.

4.3.7. Finalizing the bundle

Finalizing a bundle when in "edit" mode functions identically to finalizing a bundle when in "create" mode. See section 4.2.3.

5. Constraints and goals

5.1. Usability

Consistent with Sugar interface guidelines, we stress that use of the Bundle activity is discoverable and intuitive. The idea behind the activity—to work with a "bundle" of objects instead of strictly individual objects—is simple; so should be its interaction with the user.

6. Response to undesired events

6.1. "Foreign objects"

Archives that the user accesses via the Web or a USB drive will potentially contain objects that no application recognizes. Established Journal behavior already handles the

case that the user tries to open a foreign object, but we emphasize that any and all file extensions are permissible in bundles.

6.2. Memory

6.2.1. Extracting too much

There is a chance that there will be insufficient memory for decompressing and transferring the objects that the user selects from an archive. Unfortunately, there is no way of detecting and preventing an out-of-memory failure.

6.2.2. Including too much

In creating a bundle, the user could potentially include too many objects and make the bundle too large to save as a Journal entry. Again, we have no way of handling this.

6.3. Leaving the activity

If the user switches activities before finalizing his or her bundle, the Bundle activity should exit as if the Stop activity button was pressed—the bundle should be packed silently, and this instance of the Bundle activity should be saved to the Journal.

7. Life cycle considerations

7.1. Subsets

1. Flat view – only allow the contents of a bundle to be displayed in a flat view, making the bundle a simple "box of stuff."

2. Hierarchy – provide an option to display the contents of a bundle in either the flat view or a hierarchical view that also supports folders.

3. Selecting a single object only – adding, erasing, and extracting must be done one object at a time.

4. Selecting multiple objects – the user can select multiple objects at a time to add, erase, or extract.

5. .zip only – the only allowable archive format is .zip.

6. Three archives – include .zip, .tar, and .gz as allowable archive formats.

7. Collaboration – multiple users can interact in the same Bundle activity, and each user can make his or her own contribution to the bundle.

Note that 1, 3, and 5 are subsets of 2, 4, and 6, respectively. Subsets 1, 3, and 5 would form the most basic usable application.

7.2. Fundamental assumptions

None.

7.3. Potential changes

1. If its functionality would stay more consistent with other activities as a result, the Stop activity button could just save the state of the current instance of the Bundle activity and not actually pack the bundle. Because this change would introduce complications, the Stop activity and Pack buttons function identically for now.

2. If the change above is implemented, then leaving the Bundle activity would most likely change as well from silently packing to saving the state only.

3. The allowable formats of an archive could expand beyond .zip, .tar, and .gz.

4. We may enable the Keep button if it is deemed meaningful for this activity.

5. Subset 7 is not realistic within the scope of this project, but it is viable in future versions of the Bundle activity.

8. Glossary

8.1. Bundle

8.1.1. Disambiguation

Use of the term "bundle" is potentially ambiguous. To be clear, when we refer to a compressed collection of objects, we use lowercase ("bundle"). When we refer to the *activity*, we use capitalization and usually append the word "activity" ("Bundle" or "Bundle activity").

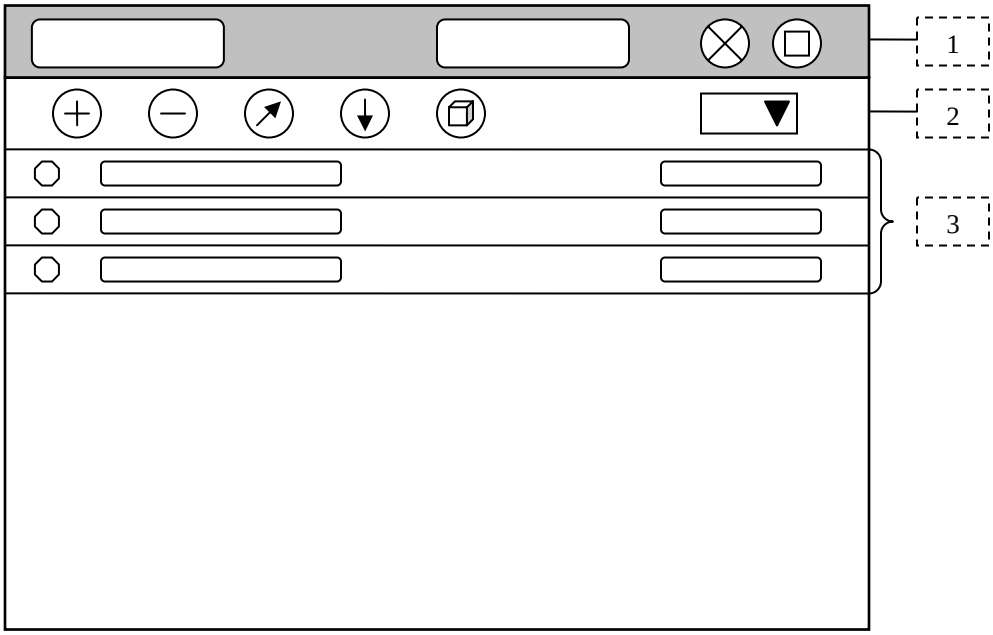
8.1.2. A bundle

A bundle can be either a bundle that was created (using the Bundle activity) by oneself or by a friend and shared, or an archive accessed from the Web or a USB drive. We use the term "bundle" for both because for software and user interface purposes, we would like the two to be identical.

9. Sources

1. "The OLPC Wiki." *Wiki.laptop.org*. 2008. One Laptop per Child association. 15 Sept 2008 <http://wiki.laptop.org/go/The_OLPC_Wiki>.

Figure 1. GUI mockup



1. *Activity bar*. Keep button is disabled for now.

2. *Bundle activity bar*. Five buttons are Add, Erase, Extract, Extract All, and Pack. Icons are to be determined. Also, there is a drop-down menu for choosing an archive format.

3. *Bundle contents*. The objects currently included in the bundle should be represented in a ListView.