OPBM

Office Productivity Benchmark

August 16, 2011

**Errata and Notes for Early OPBM Users**

By Van Smith and Rick C. Hodgin

Cossatot Analytics Laboratories, LLC.

or “Cana Labs”

**Overview**

OPBM and its GUI are still under development. There are some known issues which need be conveyed, along with their work-arounds. This document outlines the internal function of OPBM, and allows manual side-stepping of any bugs discovered or encountered within the GUI, via manual manipulation of the main XML files.

There is nothing within OPBM that has been rigidly defined for the developer panels and the format of the input files--and even those files can contain either errors or additional “meta” data within them, which are all silently ignored. This flexibility makes OPBM very powerful, but it also requires that the XML files be setup correctly for OPBM to function.

Whereas end-users will likely never see any of this as they will use the simple skinned GUI interface for execution, for script developers in the debugging phases of developing new benchmarks, it is essential and will easily facilitate new and varied releases of OPBM with little or no internal programming changes to the main opbm.jar source code, but rather only modifications to the required XML files.

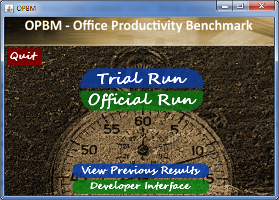
There will likely also be plugin support for new commands not currently known to OPBM.

**Two GUIs in One**

OPBM can execute with a simple, skinned GUI interface, or with the full developer interface. The desired startup option is specified by data in the settings.xml file, but can also be overridden with command line options (see the command-line document for syntax).

**Simple, Skinned GUI Concepts**

The simple, skinned GUI will have either four of five buttons on it, depending on whether or not the link to the developer interface is allowed. It looks like this:



The buttons are:

Quit - Quit the application

Trial Run - Execute a single-pass trial run for examination

Official Run - Execute a three-pass official run for benchmarking

View Previous Results - A link to allow prior results\_\*.xml files to be viewed in Results Viewer

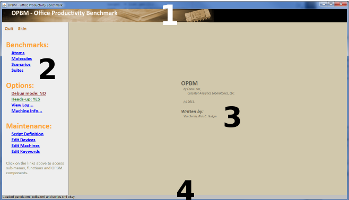
Developer Interface - (optional) Allows a link to the Developer GUI

**Developer GUI Concepts**

It is important to understand that the developer GUI is driven by underlying XML files which describe its format and structure. It is not a standard form as in most applications. But, before understanding this fully, it is important to note that the OPBM Developer GUI is divided into four main parts. These parts are updated by the contents of the XML files, which instruct what to put where:

1. Header - The “OPBM - Office Productivity Benchmark” Graphic at the top
2. Left Panel - White area along the left side (used for menu navigation)
3. Right Panel - The large tan area
4. Status Bar - The gray bar at the bottom

The OPBM Developer GUI looks like this:



The Left Panel navigation area (white area) is further divided into two main sections. The top portion contains general navigation back and forth between menus ( “Back”, “Home”, “Quit” and “Skin”). Below that are headings and contextual menu items which lead to data input screens.

To navigate into any item, position your mouse over the link and left-click. The screen will change and additional menu options and screen input items will appear.

**System Layout**

There are three primary files which drive OPBM. These ***must*** be setup correctly or OPBM will not even launch properly, but will enter an error-correcting edit mode whereby the raw XML files are loaded for editing. Outside of OPBM, these are regular XML files and can be easily edited in a text or XML editor:

1. edits.xmls - defines the way OPBM edits data on-screen
2. panels.xml - defines menu navigation options
3. scripts.xml - defines the scripts OPBM will execute

Within those files are defined the seven primary concepts OPBM employs:

1. Flows - flow control directives (for..next, if..else, etc.)
2. Abstracts - abstract commands (currently only two are defined)
3. Atoms - the smallest benchmark workload, a single executable script
4. Molecules - a grouping of Atoms into a related workload
5. Scenarios - a grouping of Molecules and Atoms into a related category
6. Suites - a grouping of Scenarios, Molecules and Atoms into a related benchmark

All XML files begin with the root tag **<opbm>** and are encoded in standard utf-8 format. The root opbm tag signifies the XML file is an OPBM file. From there, the internal tags vary by file, but their syntax should be consistent throughout each file, meaning everything in edits.xml will look like an edits.xml entry, etc.

OPBM expands special characters to a verbose explanation, such as every instance of an = appearing as (equals). This is done because Java’s W3C DOM model interprets some character combinations as control codes, and rather than wade through that quagmire of variability, OPBM simply replaces all instances of “offending” characters with their verbose counterparts.

This will make a manual edit of these files somewhat more difficult, but here is the full list of every special character that needs to be expanded (not case sensitive):

(lessthan) <

(greaterthan) >

(equals) =

(ampersand) &

(caret) ^

(percent) %

(poundsign) #

(atsign) @

(exclamationpoint) !

(tilde) ~

(singlequote) '

(doublequote) "

(hyphen) -

(plus) +

(leftbracket) [

(rightbracket) ]

(leftbrace) {

(rightbrace) }

(asterisk) \*

(backslash) \

(slash) /

(colon) :

(semicolon) ;

**edits.xml**

This file contains elements which literally define what is edited on the OPBM input screens, where and how. Its format should be very straight-forward to experienced developers. However, it does contain some new concepts which may not be familiar to even experienced developers or users, such as a dot-reference to access data within scripts.xml, a variable foreach element clause in listbox and lookupbox controls which allow for relative edits (as a single listbox can contain multiple different types of input), and the use of a pound sign prefix to signal the editing of attributes.

In addition, there are some edit fields not even defined in edits.xml. OPBM provides the ability to define “soft fields”, which are fields defined within the user-input. Those soft fields are input by typing into the fields defined within edits.xml, but exist in scripts.xml as their data. They contain a syntax like this: **{:field, prompt:}**. Once defined, they appear as relative inputs on the Atoms, Molecules, Scenarios and Suites screens within OPBM.

A careful examination of the edits.xml file should reveal its syntax and structure fairly easily, especially when observed alongside real-world use. In addition, for those instances on the Atoms screens when additional edit fields appear along the right-side with their own prompts, the use of “soft fields” should be made clear as well. Clicking on the “Zoom” button to see the original source data input of the execute command, or some of the flow control items, should also help make it clear.

All soft fields are contextual and appear and disappear as users navigate the Sequence of Operations fields lookupbox. Within the edits.xml file a single line appears which tells OPBM to update the screen in this way:

**<options relativeto="elements"/>**

All editing is handled by OPBM through the edits.xml file, and the user-defined soft fields discovered when each screen is updated. These soft fields are referred to internally as “options” by the edits.xml file.

In summary, there are no rigid edit formats provided for by OPBM. Every user input and data item defined within OPBM can be modified using this XML and user-defined soft field format. This provides an extremely flexible platform for use, making future version releases and minor changes to the user-observed GUI, easy and transparent, and often without requiring any access to the OPBM source code.

This extremely flexibility will undoubtedly require some extensive hands-on, face-to-face training to fully convey and understand.

**panels.xml**

In a similar way to the edit system, there are no hard-and-fast menus defined by OPBM. Instead, the contents of panels.xml instructs OPBM on what to display and, based on user interaction, what command to perform. The only requirement of panels.xml is a single left panel menu entry that must exist, called Main (not case sensitive). *Note: This is similar to software programs needing a main() function.*

Per this requirement, OPBM initially navigates to the Main panel. From there, the user action defines where it goes, with the requirements of panels.xml and edits.xml working hand-in-hand for traditional user input.

Certain commands are defined within OPBM to facilitate the navigation process. These can be observed by looking at the raw contents of panels.xml and edits.xml. A full explanation of all of their features would require a textbook and college-level course. However, an experienced user should be able to readily understand and manipulate the XML file directly, if new panels or edit screens are required.

An online demonstration of this can be arranged.

**scripts.xml**

This XML file contains all of the data manipulated by edits.xml. It holds not only scripts, but also all of the defined elements known to OPBM’s edit system. These include the seven types listed above, which were Flows, Abstracts, Atoms, Molecules, Scenarios and Suites. In addition, there is information which can be input or recorded about the machine being tested, its components, etc. (these features are not yet completed in OPBM, though early “hints” of their existence are present).

An examination of this file should reveal in a very straight-forward way, how its data items are stored, and how they are used internally within OPBM’s edit system for data input.

**Known Errata**

**Errata:** Listboxes and Lookupboxes. There’s a bug in the listbox and lookupbox controls which causes keyboard navigation (up/down arrows, page up/ page down) to not track properly. On-screen data items will not be updated properly, and when they are stored back to the source file (as OPBM uses a WYSIWYG input model), it results in data being stored in the wrong field, overwriting the contents of another field, etc.

**Workaround:** Use only the mouse to select listbox and lookupbox items.

**Errata:** Input Fields. There’s a bug in the “lost focus” code which will *not always* recognize a changed input text field’s contents as actually being changed. This occurs when the user navigates from a text field to a lookupbox or listbox.

**Workaround:** Click on another input box or press the TAB key to leave the field before clicking a listbox or lookupbox.

**Errata:** Menu navigation. There are time when clicking the “Home” link that the edit screen displayed on the screen will not disappear. When this happens no other edits can be displayed because they’ll be interfered with by the one which remained.

**Workaround:** Use the “Back” button, or close the app and re-launch if it occurs.

***PARIAL FIX APPLIED: August 28, 2011 code drop. All menu items that linked to non-existent entries were removed, and this fixed all instances related to that cause. One cause remains, when zero items appear in listboxes. Am working on that fix presently.***

**Errata:** Screen items not linked. There are some main menu items which are not linked to code underneath. They were created as placeholders and do not function when clicked, such as all of the Options items.

**Workaround:** None. If clicked, they just won’t do anything.

**Errata:** Un-expanded special characters. As indicated above, there are some characters stored in the XML files as special character expansions, such as = expanding to (equals). There are times when the un-expanded form may appear in various fields.

**Workaround:** None. Should not affect anything. When saved and re-loaded, will be corrected.

**Errata:** Heads-up-display position. The HUD does not always appear at the lower-right of the screen.

**Workaround:** None.

***FIXED: June 28, 2011, 2:45pm, per Jim MacDonald reporting this bug interfering with an Excel script.***

**Errata:** Heads-up-display Stop button. The HUD does not have a functioning stop button.

**Workaround:** None. If the process locks up, and the script being executed does not self-terminate through some kind of error-trapping routine (which will cause OPBM to recognize the script has exited), then the user will have to activate Task Manager and manually kill the script’s process, or (worst-case scenario) the OPBM process.

***FIXED: August 5, 2011, 10:45am. Terminate executing script by clicking the flashing green button.***

**Errata:** Spurious XML updates. There are instances where the edit system updates not only the correct item within the XML file, but also carries that update to a parent item, appending the same data.

**Workaround:** None. This introduces unused data, but should not affect anything as part of OPBM’s design is to allow previously existing items to pass silently through from XML read to XML write after update. A future update will check for this unknown data, and self-correct by removing it.

***FIXED: June 29, 2011, 09:15pm.***

**Errata:** Zero items in listbox. If the last item in a listbox is deleted, the listbox will enter an unusable state. No data can be added back in until someone manually goes into the scripts.xml file and adds a valid record.

**Workaround:**  Don’t delete the last item. If you need to delete all of the items, delete all-but-one, and then add the new item and delete the previously remaining one, or simply set the value in that last item to something like “Unused”. It can even contain invalid information, but with this OPBM release it just can’t be blank.

**Undiscovered Errata**

There are undoubtedly other undiscovered bugs. And there are several enhancements currently planned in our internal Bugzilla database. Until a public Bugzilla website is created, please send all bugs to:

Van Smith, [van@canalabs.com](mailto:van@canalabs.com)

Rick C. Hodgin, [rick@canalabs.com](mailto:rick@canalabs.com)

Zero-day and critical service releases will be issued promptly after show-stopper bugs are discovered.  
Regular service releases will be scheduled periodically (most likely once every week or two).

Please report all zero-day and critical bugs to:

Rick C. Hodgin

[rick@canalabs.com](mailto:rick@canalabs.com)

317-879-6374

Refer to the **opbm\src\resources\** and **opbm\src\resources\xmls\** for access to:

panels.xml

edits.xml

scripts.xml

settings.xml