Appendix A

Updates for VisualWorks 7.3.1

Overview

VisualWorks 7.3.1 is a “patch” release, consisting mainly of fixes to problems we found with 7.3. There are also a couple of features being moved from preview into full supported product.

With only a few exceptions, documentation has not been updated for this release. These release notes cover the essential changes and additions.

For late-breaking information on VisualWorks, check the Cincom Smalltalk website at http://www.cincom.com/smalltalk. For a growing collection of recent, trouble-shooting tips, visit http://www.cincomsmalltalk.com:8080/CincomSmalltalkWiki/Trouble+Shooter.

ARs Resolved in this Release

The Action Requests (ARs) resolved in this release are listed in: fixed_ars.txt in the **`doc/`** directory.

Additional ARs may be discussed in individual sections of these release notes.

Outstanding ARs and limitations are noted throughout these release notes, as appropriate.

VisualWorks Base

Printing

VisualWorks now by default supports PostScript Level 3 printing. With Level 3 PostScript printing VisualWorks can finally print paints that implement a Pattern. Opaque images with high color content are now rendered orders of magnitude faster.

All PostScript printers sold since 1997 support PostScript Level 3. In the unlikely event your PostScript printer does not support Level 3 you may reduce the implementation level VisualWorks writes PostScript in on the PostScript Options page of the Settings Tool (to open this tool, select **System ⟶ Settings** in the Launcher window).

Exiting VisualWorks on Unix SIGTERM

Unix provides an array of signals. SIGTERM is a signal that by convention is used to inform a process that it is about to be killed. The intention is that the process can then do any finalization activities that it requires (closing sockets, saving data, etc.). However, unlike SIGKILL (SIGKILL is always signal 9, as in 'kill -9 <pid>'), an application is allowed to ignore SIGTERM.

For all supported Unix systems, in order to install a SIGTERM handler you must first ensure that

```
ObjectMemory objectRegisteredWithEngineFor: 'acceptQuitEvents'
```

returns false. This is the case in a headless image, so nothing more needs to be done. Just register your semaphore for SIGTERM and off you go.

In a headful image, InputState>>run registers acceptQuitEvents as true. This informs the VM that if a SIGTERM signal is recieved it should be forwarded to the image so that the image can shut down. If you wish to handle SIGTERM events in a headful image, you need to set acceptQuitEvents to false after the initial system startup has completed.

The input state process is started each time the image is started, and it will always set acceptQuitEvents to true. You need to respond to this accordingly. This is best done as an action in the UserApplication class, or your application-specific subclass, which is started after WindowingSystem starts up, and is where InputState>>run is done. All of this is just to stop the image from intercepting SIGTERM.

To catch SIGTERM in a headful image, do the following:

1. Start the image with the command line option:

```
-doit "ObjectMemory registerObject: false
  withEngineFor: 'acceptQuitEvents'"
```

2. Create a subclass of UserApplication that defines a method main as:

```
main
    "Tell VM that we want to receive SIGTERM events."
    |sem|
    OSHandle currentOS == #unix ifTrue: [
        ObjectMemory registerObject: false
            withEngineFor: 'acceptQuitEvents'].
```

3. Register to have a message sent to one of your classes in which the actions in (2) are performed. To do this, evaluate the following in a workspace and snapshot your image:

```
SystemEventInterest
    atSystemEvent: #returnFromSnapshot
    send: #setupTermHandler
    to: MyClass.
```

4. As a *last resort*, execute the following code in a workspace and snapshot your image:

```
|idx|
idx := InputState.EventDispatchTable identityIndexOf:
        #send:eventQuitSystem:.
idx > 0 ifTrue: [ InputState.EventDispatchTable at: idx put: nil ].
```

   (Keep in mind that if you chose this mechanism you will be disabling the reception of quit events on ALL platforms that the image runs on.)

SIGTERM is the only signal VisualWorks passes to the image by default.

Tile Phase Corrected

GraphicsContext>>tilePhase: is dependent upon object engine implementation. 7.3.1 engines correct two problems:

- tilePhase: was not previously implemented on the MacOSX native engine. It is now implemented in the 7.3.1 engine.
- tilePhase: was implemented incorrectly in all Windows engines. It is not implemented correctly.

tilePhase was already implemented correctly on all X11 engines.

Because of errors on Windows platforms, users of tilePhase: on Windows would have had to write code like this to correct for errors:

```
aPointWhereTheyWantedTilePhaseToBeOffset negated.
```

or

```
(aPointWhereTheyWantedTilePhaseToBeOffset // 2) negated.
```

(This code, if run on X11, would be offset incorrectly.)

If you implemented such work-arounds, you need to replace this code. With the 7.3.1 engines you have to specify the actual offset. This now works on all platforms.

Advanced Tools

Profiling

(Republished from 7.1 Release Notes)

The ATProfiling parcel in **`advanced/`** has been replaced by two new parcels, ATProfilingCore and ATProfilingUI. These new parcels include the functionality previously shipped as ATProfilingEnhancements in the **`goodies/`** directory. They also segregate profiling functionality into "core" and "user interface" components. This is a prelude to shipping "attach-and-profile" and "distributed process" profiling in some later release. Only the "core" components needs to be loaded into a remotely profiled, and possibly headless, image.

The new parcels support single-process as well as multi-process profiling. Users are urged to remember that all the profilers rely upon a statistical sampling heuristic to estimate, rather than on instrumentation to directly measure, the resources consumed by a process. Multiprocess profilers distribute the probes used to estimate resource consumption over several processes, rather than one, and the distribution may be uneven. Also, running multiprocess profilers does cause garbage collection and other maintenance processes to run more frequently than otherwise. These facts should be kept firmly in view when setting up multiprocess profiling runs and when estimating the reliability of their results. Within these limitations, multiprocess profilers have proven useful in tuning web applications involving many hundreds of processes.

In these new parcels, the pre-existing public profiling API has been preserved. The primitive lists have been revised. The Profile Outline Browser is no longer limited to a maximum of three reports. The profiling user interfaces, by default, now open showing new advisory text and multiple examples. To make these user interfaces show code templates instead, evaluate

```
Profiler showTemplates: true
```

The old profiling parcel, ATProfiling is still shipped in the **obsolete/** directory, because other obsolete and preview parcels list it as a prerequisite. If appropriate, users should explicitly update the prerequisites of their parcels to list the new profiling parcels rather than the old one. Note that the old ATProfiling parcel is incompatible with the new set of parcels, and vice versa. They should not both be loaded into the same image.

Note also that the *Advanced Tools User's Guide* has not been updated for this change in time for release.

DLLCC

Windows VM requirements

As of VW 7.2, **visual.exe** does not support the full range of DLLCC facilities any more.

On Microsoft Windows platforms, a crash may occur when using the **visual.exe** VM with DLLCC. This because **visual.exe** does not use the **vwntoe.dll**, but is linked statically.

The reason is, that .exe files don't export their symbols. Thus a DLL, even if it was launched from the image+VM, cannot access VM functions. But whether the VM is dynamically linked as a DLL, with its functions exported, it can be accessed from other DLLs.

The **vwnt.exe** VM does use the **vwntoe.dll**, which is a main part of the VM, and thus provides the required possibility to access VM functions from a DLL.

Solution: Use the **vwnt.exe** instead of **visual.exe** .

GUI Development

Over 50 GUI related bug fixes are incorporated in VW 7.3.1. The highlights of these fixes are summarized below.

Windows Multi-Monitor Support

The VisualWorks GUI may now operates seemlessly on two or more monitors on Microsoft Windows installations that support them. There is no need to use the MultiMonitor goodie of previous releases to open, drag, or use windows on more than one monitor.

Optional UI Compatibility Parcels

Two parcels are provided in this release, for optional installation, that change the traditional VisualWorks menu and text editor copy behavior to be more platform faithful: MenuUICompatibility and CopyBufferUICompatibility.

Although a majority of VisualWorks customers have requested these features, they introduce subtle changes to UI behavior that some users have become accustomed to in VisualWorks over the years. Therefore, we have not changed the default UI behavior at this time. If you want the improved, platform faithful menus and text editor copy function, load these new parcels into your image.

Available originally as preview in VW7.3, the MenuUICompatibility parcel addresses incompatibilities of VW menu bar menus with Mac, Windows, and Motif menu requirements. Issues addressed include:

- On Windows, a mouse drag (i.e., mouse button hold and move) to a menu item with a submenu should not close the menu or submenu upon button release.
- On Windows only, menus highlight disabled menu items; other platforms skip past disabled menu items entirely.
- MacOSX menus do not wrap highlight of menu items for key up/down navigation.
- Mac OS8/9 closes all menus upon any key press.
- Unix platforms do not highlight menu items while moving the mouse cursor over an item; the mouse button must be down to do this.
- Except on Unix, submenu item menus open after about a 0.5 second delay when the mouse cursor is over them.
- For the Motif and Windows look, a menu and submenu should remain open after a mouse click and release on a menu item with a submenu.
- On Mac platforms, moving the mouse outside a menu to the right should not close all submenus open.
- On Mac OSX, <Tab> and <Shift><Tab> navigate and open menus right and left in the menu bar.
- On Mac OSX, a menu and all its submenus should not close prematurely if the up or down arrow keys are used to navigate to a menu item with an unopened submenu.

- For MS Windows and Motif menus, a menu item that opens a submenu cannot become a selection itself and then close; only a menu item without submenu may be the menu selection.

The CopyBufferUICompatibility parcel modifies VisualWorks so that the contents of the copy buffer is not altered when a <Backspace> or <Delete> key is pressed to remove selected text in a text editor. The legacy VisualWorks text editor operation for a <Backspace> press cuts any selection from a text editor and places it in the copy buffer; a <Delete> key press will empty the copy buffer. CopyBufferUICompatibility fixes the VisualWorks copy buffer behavior to be equivalent with most text editors outside of VisualWorks.

Drag and Drop List Cues

In VW7.3.1, List and TreeView widgets may now be customized to show drop feedback upon insertion between elements, element replacement, or both. Send the following variations of showDropFeedbackIn:allowScrolling: for the visual feedback desired:

showReplaceDropFeedbackIn:allowScrolling:
Show a box about the item the object is to be dropped on.

showInsertDropFeedbackIn:allowScrolling:
Show a line between two items where the object is to be inserted.

showInsertReplaceDropFeedbackIn:allowScrolling:
If the cursor is centered over an item show a box about it; otherwise show a line between the two items nearest the cursor.

When using drop feedback for insertion, the view targetIndex will appear in increments of ½ indicating that the drop occurred between items.

Two new example parcels, DragDropList and DragDropTree, demonstrate drop insert and replace to List and TreeView widgets, respectively.

Windows Look Policy and Fonts

The auto-select look on Windows 2003 is now the Windows 2000 look as it should be. The default system font family requested for the Windows XP look is now Tahoma. For MS Windows, the number of fonts identified as serif, sans serif, and fixed by VisualWorks has been increased to account for the larger set of fonts distributed with the current Windows XP and 2003 releases. This improves font matching when these attributes are specified. Finally, the size and family of the default system font used by VisualWorks should no longer depend highly on the MS Windows user preference to only use TrueType fonts or on the number of fonts installed.

Named Font Selection

Filing out or loading Named Font definitions no longer cause exceptions if the font name includes a blank or if it defines a system font.

Editable ComposedTextView

Since VW 5i.2, windows created by ComposedTextView instance creation methods have been read-only. In 7.3.1, new methods have been added to create an editable window on a ComposedTextView instance. They are:

Read-only (original)	Editable (new)
open	edit
open:label:	edit:label:
open:label:icon:	edit:label:icon:
open:label:icon:extent:	edit:label:icon:extent:
createOn:label:icon:	editOn:label:icon:

TreeView

If a TreeView appears in a Tab Control, its text emphasis will now appear. Expanding and then contracting a node in a single-select TreeView will no longer remove the node as a selection.

UI Painter

When several UIPainter canvases are open, closing one canvas now raises the next canvas immediately, brings it to focus, and updates GUI Painter Tool contents accordingly.

Dataset

Tabbing to another cell in a dataset no longer unexpectedly toggles a cell's check box.

Dialogs

Dialogs on Windows platforms previously reduced their height by 8 pixels the first time they were moved. In 7.3.1, dialogs on Windows platforms now maintain a constant size.

Menus

The menu pragma computedSubmenu:nameKey:menu:position: is now consistent with other menu pragmas, in that the nameKey: argument names the submenu added to the menu whose access path is given by the argument to menu:. The MenuPragma2 parcel contains a revised example that uses it.

Internationalization

Parcels in the **`japanese/`** directory contain a Japanese locale, plus additional parcels that add changes to other parts of the system in order to make them more Japanese-friendly. For example, if you are using the UIPainter and Japanese parcels, you would probably want to load JaUIPainter.

Currently, only Solaris, HPUX, Linux, and Windows support this Japanese locale. Support for Mac Classic and Mac OSX is planned but is not yet available.

This locale is known not to coexist well with the Unicode parcels. Over time, the conflicts between the two bodies of code will be resolved, but for now, you may want to avoid trying to use both in the same image.

Web Services

Refactored Parcels

(AR#48432) The BindingTool parcel was split in two parcels: XMLSchemaMapping and XMLObjectBindingTool. To avoid conflict with old classes, be sure to install 7.3.1 WebServices support in an empty webservices directory.

New XML-Object Binding Wizard

(AR#48506) The XML to Object binding wizard help you create XML to object binding from specified classes.

Application Server

Reduced large space multiplier

In recent versions, the size of large space was greatly increased to improve performance under heavy load. The primary factor in this was the rapid allocation and release of socket buffers. This improved load performance, but required setting aside a large chunk of memory which was otherwise unusable. Version 7.3 introduced a mechanism for re-using socket buffers that greatly reduces this load, and so the increased large space is no longer necessary and it has been reduced to the normal size.

VisualWave labels with images and using https

In VisualWave, if a widget uses an image as its label, and the image was obtained over https rather than ordinary http, an error would result. Also, the image was not visible in normal editing. Note that using https, a variety of additional security-related exceptions may occur, and need to be dealt with appropriately by user code.

Allow Servers to serve one particular interface

If a server has more than one network address, previous versions automatically bound to all of them. This release provides a "bind to all interfaces" setting when creating a server. If this is true, then behaviour is the same as in previous versions. If not, it will bind only to the named interface. For example, if the server name is 'localhost', only request from the same machine will be served.

Remove demos from prebuilt runtime.im

The **`runtime.im`** supplied in the **`$(VISUALWORKS)/web/runtime.im`** previously included a number of the examples included under the **`$(VISUALWORKS)/web`** directory. This made it work reasonably well as a demo image, but was inappropriate for use as a production image with parcels loaded into it at startup. These examples are no longer included in the image.

Runtime Packager

Error log in UTF-8

The runtime packager error log was previously saved in the default encoding for the platform. However, if characters outside that encoding range appeared anywhere in the stack trace, the entire logging operation would fail. The error log has now been switched to be in UTF-8 encoding, so that all characters can be represented. Note that in viewing the error log, you may need to adjust your text editor to ensure it is using the correct encoding.

Loading old Runtime Packager Parameters

When loading runtime packager parameters from previous versions, packaging operations could fail. One cause of this was that the Subsystem hierarchy, introduced in version 7.3, was not included in the previous specifications. When reading a set of parameters from an older version, we now make sure that the default kept classes and methods for that method are included. If you are aggressively trimming an image, this means that you will want to review the kept and deleted classes to ensure that it is not keeping more than you intended.

Runtime Packager command-line options

In version 7.3, many of the Runtime Packager command-line options (e.g. **`-pcl`**, **`-cnf`**) duplicated operations that are now part of the base image. If both sets were activated, this could result in the option running twice. This release removes the Runtime Packager mechanism, and extends the base image mechanism to cover all of Runtime Packager's options. Whether or not these options run is now controlled from the base image settings, not from Runtime Packager's.

Windowless runtime image would fail to exit

In version 7.3, if a runtime image was set to exit on last window closure but the image did not open a window at startup, the image would fail to exit. A windowless runtime image set to exit on last window close should now complete any task assigned at startup and exit.

Fix overwriting of memory sizes

A bug in runtime packager caused the memory sizes at startup to be reset to the defaults if they had been set in the image before running Runtime Packager and not explicitly set in the Runtime Packager tool.

Preview

Standard IO Streams

There is a new package, StandardIOStreams, that defines three shared variables in the OS namespace: Stdout, Stdin and Stderr. These variables hold streams on the three standard i/o streams: stdout, stdin and stderr.

This package does not work with any version of VisualWorks before 7.3.

Once you have loaded this package, you need to save and restart the image. If you are running on Windows, be sure that you are using **`vwntconsole.exe`**; otherwise, the handles cannot be created.

To write to Stdout do something like:

```
OS.Stdout nextPutAll: Timestamp now printString; cr
```

or the following:

```
OS.Stdout lockWhile:
    [OS.Stdout
        nextPutAll: 'testing';
        cr;
        nextPutAll: 'testing';
        cr;
        nextPutAll: '123' ; cr]
```

Stdin can be read like this (on Windows, press CTRL-Z in the console to end the stream):

```
[Stdin atEnd] whileFalse:
    [Stdout nextPut: Stdin next asUppercase]
```

MQ-Interface

A new preview package has been included, which provides an interface for WebSphere MQ using the shared libraries provided by IBM. WebSphere MQ is a message base communication system by which two or more application can exchange messages through a queue. This package allows you to access this API from a VisualWorks application.

Refer to MQ API.doc for information.

Pollock

The Pollock preview has been updated.

Note that a few simple examples are available in the open and the vw-dev repositories. Load the Pollock-Examples bundle.

Also, initial drafts of two chapters, on building a GUI interface programmatically in Pollock and on interfacing to a domain, are included with this release, in the **`preview/Pollock/doc`** subdirectory.

Also, Sam posts suggestions in his blog:

http://www.cincomsmalltalk.com/userblogs/pollock/blogView

Glorp

Glorp is a third-party, open-source project that is provided under terms specified by LGPL. Refer to the file COPYING.TXT for the full license.

This release includes an updated version of Glorp, corresponding to version 0.3.62 in the public Store repository. This has many new features since version 7.2. This is still a preview and is unsupported and missing many important features. Among the most significant additions in this version are:

- optimistic locking
- composite keys
- resolving insert order at the row level, rather than just the table level
- filtered reads (query optimization)
- queries can now return cursors
- additional cache policies, including a timed proxy mechanism that's quite neat.
- support for Oracle array binding on insert and for grouping multiple statements together on a line for databases without array binding.
- pre-allocation of sequence numbers. For databases that don't use identity columns we can get all the sequence values we need in one go rather than one-by-one.
- renamed "criteria" to "whereClause" in queries and "mappingCriteria" to "join" in mappings.
- renamed PrimaryKeyExpression to Join
- allowed any kind of mapping to use a link table. This makes the one-to-many/many-to-many distinction obsolete. Most methods on those classes have been moved to the superclass.
- added subselects, created using anySatisfy: or noneSatisfy:.
- minimal (absolutely minimal) mutual exclusion on query execution

- database-specific functions
- the ability to use functions in more places
- added unionAll: and minus:/except: on queries
- support for mapping to "imaginary" tables, where an object can be defined by the existence of one or more joins (e.g. StoreClassExtension)
- support for mapping to a group of rows (e.g. StoreVersionlessPundle).
- changes to VW proxies to make them both transparent and debuggable, a tricky business
- removed some bug fixes to Oracle that should now be fixed in VW 7.2.1.
- added a new layer of meta-description, the ClassDescription. This gives us a formal way of modelling the classes, rather than having that information be implied in the mappings. This means that the code to create most mappings is greatly reduced.
- changed the way mappings are created and initialized to use this information. Note that the old syntax is still supported.
- direct comparisons to objects now supported. e.g. where: [:each | each = anObject], or where: [:each | each thing isNIL]. (Note that isNil may be optimized away).
- basic mapping of one form of dictionary
- changes to the argument block of ad hoc mappings
- allow retrieve: of a to-many relationship
- allow mappings to pseudo-variables. These can be used to simplify queries, but don't actually read anything.
- added glorp-specific exception classes (a couple)
- changes for compatibility with GNU Smalltalk
- some improvements (though not enough yet) to blob and clob types.
- various bug fixes and performance enhancements

Many thanks to the various contributors. Particular thanks in this release to David Pennell for optimistic locking, Michael Lucas-Smith and Anthony Lander for the timed proxies, Andrei Sobchuck, Boris Popov, Radoslav Hodnicak and Andre Tibben for bug finding, fixing, and feature

suggestions. Victor Metelista and Anthony Boris for work on DB2 and VisualAge that isn't integrated here yet. Paolo Bonzini for porting to GNU Smalltalk, and many others for their contributions.

Faster Store Replication

The StoreForGlorpVWUI parcel provides a user interface for replicating packages and bundles between Store repositories. This is based on a Glorp schema for mapping a Store repository. It provides some advantages over the existing StoreReplication goodie, the primary one being that it is much faster in most circumstances, particularly against a remote repository. In addition, it provides some fairly sophisticated filtering of the things to be replicated and runs in the background rather than tying up the primary Store connection.

> **Caution:** This is unsupported, prerelease software. It has so far been tested only against Oracle and Postgres repositories and is known not to work on SQL Server due to case sensitivity issues. An error in this code could seriously damage a Store repository. Be extremely careful, and make sure that your repositories are well backed-up.

XML Schema Support

The release note (7.0) that XSchema was promoted to product at that time was premature, and it remains in preview. To use XML schema support, load the XSchema parcel, **`preview/parcels/XSchema.pcl`**.

OpentalkCORBA

The main improvement in 7.3.1 is added support for character code set negotiation. This finally allows communication of characters outside of the western ISO8859-1 encoding range. It also enables true interoperable support for the **`wstring`** and **`wchar`** IDL types. The marshaling machinery preserves the efficiency of ISO8859-1 string marshaling, bypassing the encoding overhead.

IIOP brokers now advertise supported code sets in the IORs they produce. Corresponding IOR components can be configured via new CDRMarshalerConfiguration parameters (#codeSets and #orbType). There is now also a "null" IOPCodeSets to allow configuring broker code sets as unspecified rather than whatever is the current global default. The default code sets advertised by IIOP brokers is now full set of supported encodings, see IOPCodeSets class>>default. It might be desirable to restrict that to smaller set in some cases. The code set negotiation algorithm is

fully implemented with the exception of the code set compatibility test. Consequently, if there is no match between client and server code sets, the algorithm could possibly pick an incompatible code set for transmission. If that happens you'll need to reconfigure the broker code sets to avoid this situation for now.

There have been a number of smaller changes and fixes. Here are the most interesting ones:

- added narrowing and widenning API to IORs and RemoteObjects
- added RemoteObject>>_release to allow for explict and immediate release of the proxy from the broker cache, rather than waiting for the GC to kick in.
- added #nameService convenience API to Broker
- IIOPObjRef>>oid is now enforced to be always a positive integer (to support OID autogeneration and to save some space); consequently there are new API variants (#oidBytes) for IIOPObjRef creation, to support IIOP object keys and to continue supporting specification of oids as byte objects (string, symbol, byte array)
- fixed a bug in marshaling of GIOPCloseConnection, which caused the connection closure handshake to fail and yielded a read error in the server process upon connection closure
- fixed a problem causing the broker create a new connection for every request, instead of reusing previously established connection

Known Limitations

Unicode preview

Unicode support for Windows remains in preview. Note that this support requires Windows 2000 or later, and works best on Windows XP. This limitation was not noted previously.