A

Updates for VW 7.7.1

VisualWorks 7.7.1 is a “patch” release, consisting mainly of fixes to problems we have found or have been reported with 7.7. There are also several feature enhancements.

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

These notes comprise an appendix to the VisualWorks 7.7 Release Notes. For late-breaking information on VisualWorks, check the Cincom Smalltalk website at http://www.cincom.com/smalltalk.

Installation

64-bit Installers

There are now installers built for the supported 64-bit platforms, rather than needing to use 32-bit compatibility mode to install.

Virtual Machine

Upgraded zlib

The VMs for Windows, AIX, HPUX and Solaris include an updated version of zlib (1.2.5).

Compiled code cache compaction statistics

With this release, a new instance variable called numCompactNMethods has been added to ObjectMemory. This variable reports the number of times that the compiled code cache was compacted due to a compiled code cache allocation failure.

When a compaction occurs, some native methods are discarded to make room in the compiled code cache. In this regard, compiled code cache compactions can be seen in terms of cache spills. The compaction operations, or spills, dump native methods irrespective of how frequently they have executed. Thus, spills may result in some virtual methods being translated into native code again. While a few compiled code cache compactions are acceptable, the use case for this counter is to detect situations that result in excessive compactions.

You may increase the size of the compiled code cache by using the sizesAtStartup: message implemented on the class side of ObjectMemory. As always, measure the effect on your application before and after the change before applying the change permanently.

The Advanced Tools profilers have been updated to report the number of compiled code cache compactions. For the sake of brevity, the label used is "JIT cache spills".

32-bit file activity on 64-bit Windows

32-bit VM file activity is redirected by default on 64 bit Windows.

If you will run a 32-bit VM on 64-bit Windows, you should be aware that certain folders in the file system will be redirected by Windows. The 32-bit VM cannot guess whether the application's attempt to list the contents of a directory, or to open a file, should be redirected or not. Unfortunately, moreover, the functions that turn redirection on and off are only available on 64-bit Windows. Thus, it is not suitable to include calls to said functions in 32-bit VMs since these VMs are intended primarily for 32-bit Windows. Consequently, applications may need to manage file system redirection themselves in rare cases. See the following MSDN articles for more information.

- http://msdn.microsoft.com/en-us/library/aa364418%28VS.85%29.aspx
- http://msdn.microsoft.com/en-us/library/aa365743%28VS.85%29.aspx

Zero Extent for Pixmaps and Masks

Primitives 900 and 901 no longer accept zero extent arguments on any platform.

General MacOS X Performance

In previous versions we had to compile the MacOS X virtual machine at a lower optimization level to work around problems that emerged when compiled with full optimization. This is no longer necessary, so the VMs are compiled with a higher optimization level (-O2) which should yield improved general performance.

MacOS X Bitmap Performance

In particular with MacOS X there were performance issues with large bitmaps, particularly noticeable with the use of double buffering window policies. The bitmap display has been significantly changed, yielding much better performance.

Large File Support on AIX

The virtual machine is now compiled with support for large files (more than 2GB) on AIX.

Base Image

Date Creation Changes

Changed in 7.7 but without a release note, Date creation methods no longer interpret two-digit years as designating a year in the current century. Full four-digit years are now required.

Changes to the VI's memory management policy

The image side memory manager has been substantially improved in this release. There are two main types of modifications.

First, several modes of misbehavior have been identified and removed. For example, under some circumstances, it was possible for the memory policy to claim there was a memory emergency when in fact there would have been plenty of memory if a GC had been invoked. Moreover, in some cases, it was possible for the memory policy to assume there was plenty of memory available when in reality memory was running dangerously low. These were aggravated by the fact that changing sizesAtStartup: would have no effect on the current memory policy. In addition, memory policies were not 64 bit aware. Finally, memory policy classes no longer have shared variables, since changing shared variables on a running memory policy may have detrimental consequences.

Second, multiple changes were introduced to improve application performance. In general terms, the way several memory management boundaries and quantities are calculated were changed from being a hard coded number to a calculation based on the memory upper bound and the image type (32 or 64 bits). When considering the implications to the preferred growth increment, this value has gone from 1 million bytes to roughly 22 megabytes (assuming a 512 megabyte memory upper bound). The net effect is that, as the image grows, less GC operations will be performed each time a new old segment is allocated. Moreover, the growth regime upper bound has been increased as well and, thus, the IGC will have better chances of keeping up with old space allocations before regular GCs are invoked. The free memory upper bound has also been increased so that images do not get into a cycle of growth, GC, allocation of a new old segment, deallocation of the new old segment due to some objects becoming garbage, followed by growth, GC, allocation of a new old segment, and so on. Finally, larger old segments have two beneficial side effects. First, they help prevent memory fragmentation problems that can cause allocation failures when, despite an abundance of available memory, there is a shortage of contiguous available memory. Second, the compacting GC currently runs considerably faster with less segments (compare >400 old segments in a 500mb image with a 1 million bytes preferred growth increment vs. ~20 old segments with a 22mb preferred growth increment).

Some of these changes represent significant behavior changes with respect to the previous release. For this reason, the class MemoryPolicy has been updated to correct known misbehaviors and to remove the shared variables. Nevertheless, to fully benefit from this work, two new memory policy classes have been introduced to the system: MediumGrainMemoryPolicy, and LargeGrainMemoryPolicy. They represent two levels of memory policy tuning aggressiveness, and they should also serve as an example of how to introduce new memory policy classes easily. During this work, at one point we observed that our memory policy stress tests ran in 2.15 hours with the legacy memory policy, 1.15 hours with the

MediumGrainMemoryPolicy, and 15 minutes with the LargeGrainMemoryPolicy. Thus, VisualWorks comes with the LargeGrainMemoryPolicy installed as a default.

If your application uses a customized version of the class MemoryPolicy, you can still use it after integrating your changes and taking into account that the MemoryPolicy class no longer defines shared variables. Thus, the memoryUpperBound should be set with the corresponding class instance variable accessors. If the free memory upper bound or the growth regime upper bound were customized, then simply make a subclass of MemoryPolicy called, e.g., ApplicationMemoryPolicy, and refine the instance initialize method as follows:

```
initialize
   super initialize.
   self freeMemoryUpperBound: someNewValue.
   self growthRegimeUpperBound: someNewValue
```

To install a memory policy class, simply evaluate

```
GivenMemoryPolicyClass install
```

Note that, if you have a memory policy class subclassed from Object, you should change it to subclass AbstractMemoryPolicy and ensure that all the protocol defined in terms of subclassResponsibility is implemented appropriately.

Those interested in creating new memory policy classes should keep in mind the following resources:

- Memory Monitor. This is a graphical memory monitor originally written by John Brant and Don Roberts. The version shipped in the distribution has been enhanced to make it more useful. Besides displaying more and better quality data, it also has the capability to write CSV logs for offline analysis.
- MemoryPolicyChecker. This package contains SUnit tests that ensure memory policy working assumptions are not violated. These tests are an invaluable tool that will detect often subtle interactions between the many memory policy variables.
- MemoryPolicyTuner. This package offers memory policy suggestions based on the image's current configuration.
- MemoryPolicyStressTest. This package contains tests designed to verify the performance of the current memory policy, in regards to both reliability and execution speed.

The memory management documentation has been significantly revised and enhanced as well. Pay special attention to vwMemoryManagement.pdf in the TechNotes folder.

Finally, the default sizesAtStartup multipliers have been updated as follows:

- Eden and survivor semispace multipliers increased to 5x.
- Large space multiplier increased to 5x.
- Native stack multiplier increased to 20x.
- Compiled code cache multiplier increased to 3x.
- Old space headroom increased to 7x.

These values, just as setting all multipliers to 1.0 as in previous releases, represent a compromise between memory usage and performance. As always, you should measure the impact of changing these values in your application.

Myers Diff Algorithm

VisualWorks 7.7.1 includes a Smalltalk adaptation of the popular difference algorithm as first defined by Myers' solution to SES/LCS with the Hirschberg linear space refinement as described in the following publication:

> E. Myers, "An O(ND) Difference Algorithm and Its Variations," Algorithmica 1, 2 (1986), 251-266.
> http://www.cs.arizona.edu/people/gene/PAPERS/diff.ps

This is the same core algorithm used by GNU diff(1), svn, and other tools.

The easiest way to use this algorithm is with the new differences: method added to SequenceableCollection. E.g.,

```
'human' differences: 'chimpanzee'
```

The result is termed the "shortest edit script," which is an Array of SubsequenceDifference subclasses (SubsequenceMatch, SubsequenceInsert, and SubsequenceDelete) which describe the sequence of steps taken to turn the original sequence ('human') to the revised sequence ('chimpanzee'). In the example, we end up with the following edit script: Insert c, copy h, delete u, insert i, copy m, insert p, copy a and n, insert z and e and e.

In addition to using the helper method on SequenceableCollection, one may create a SequenceableCollectionDifferences directly. This gives the added control of setting the comparison function. For example:

```
diff := SequenceableCollectionDifferences new.
diff comparisonFunction: [:a :b | a asLowercase = b asLowercase].
diff differencesFrom: 'Human' to: 'Chimpanzee'
```

Which would create the same result as the first example, but indifferent to case.

While comparing strings is the obvious use case for this algorithm, it should be noted that it is not limited to string comparison. It can be used for DNA symbols, strings, numbers, whatever one can think of to put in a SequenceableCollection. Long before the new comparison tool found in 7.7.1 gets down to showing source differences, it has used the differences: algorithm on multiple kinds of object, and in a number of different passes to get the point of which source code fragments to look at.

Behavor allSubclasses now Depth First

Behavior allSubclasses was found to differ from it’s allSubclassesDo: counterpart. They are both the same now, depth first, rather than breadth first.

ClassOrganizer defaultProtocol no longer ambiguous

For a long time, this message was used for two different purposes: to actually put it under the default protocol, or to indicate no protocol. This ambiguity causes confusion with code control systems such as Store. It is now possible to use nil as the second argument to the ClassDescription compile: aText classified: aCategory method. Where appropriate, it is now used, to indicate that the categorization of the new method is indeed undefined.

Zero extent Pixmaps and Masks

Prior to VisualWorks 7.7.1, zero extent Pimaps and Masks have been handled in an inconsistent way. On MacOSX, they were modeled in the virtual machine with 0.1 extent surface, which could be done with NSImage objects. X11 handled them by creating an invalid handle, and then continuing to work apparently, but output messages about invalid handles to stderr. Windows libraries would actually take a zero extent surface, and pad it up to at least 1.

In VisualWorks 7.7.1, because of these various differences, the virtual machine has been changed to uniformly disallow allocation of Masks or Pixmaps of zero extent. The allocation primitives will now return a primitiveFailure if either dimension of the extent is zero or less.

To continue to support zero extent Pixmaps and Masks, a scheme similar to that employed by the Windows library is used, but in the Smalltalk image level code. The methods which call the allocation primitives have been changed to force the allocated size to have a minimum dimension of 1. The Smalltalk Pixmap and Mask objects which model these surfaces will still have the zero based sizes, but the handle behind them will have at least one pixel.

Application programmers shouldn't see any ill effects of this, other than spurious error messages may go away on X11, and the behavior is now consistent across platforms.

Window isEventDriven: removed

This method has had no effect for some time, as all Window objects are always event driven now, and for many releases. It was removed.

System Time Zone

VisualWorks 7.7 introduced the SystemTimeZone class, which delegates TimeZone related operations to the OS. Its main advantage is that it allows an application to automatically reflect the OS settings, without the need to set up an equivalent internal TimeZone. However the OS facilities often support only a limited range of valid timestamps (e.g.: 32-bit Unix: 1900-2038, Windows: 1600 - 30827), which many customers found too limiting.

In this release we're introducing ExtrapolatedSystemTimeZone, which extends the SystemTimeZone to extrapolate results for timestamps and second counts that are out of the OS supported range in following way:

1. for second to/from timestamp conversions it falls back on the internal UTC implementation (TimeZone null)
2. for local to/from universal timezone conversions it extrapolates the result of equivalent conversion of a timestamp from a year on the corresponding end of the OS supported range, e.g. to convert Timestamp from year 1111 on 32-bit Unix it will convert similar timestamp with the year set to 1902 and return the same result with the year set back to 1111.

This extrapolation method should be acceptable for most applications and therefore the default TimeZone has been changed from SystemTimeZone to ExtrapolatedSystemTimeZone. Applications, for which the extrapolation method is not appropriate, can switch back to SystemTimeZone or implement different method using the existing classes as examples.

Reorganization of Subsystems

In this release, we have reorganized the SubSystem classes used during system startup and shutdown. One of the changes is to group them into four phases. These are:

Core

Kernel

Basic

Runtime

Subsystems that initialize kernel classes in some fashion (e.g., Timers, Locales, StandardIO) happen during the kernel phase. Basic services (e.g., Windowing, Headless) happen during the basic phase. The phases are delimited by new subsystems. So the core phase is everything that happens between CoreSystem and KernelSystem. The kernel phase happens between KernelSystem and BasicSystem. Then the basic phase happens between BasicSystem and RuntimeSystem. And the RuntimeSystem indicates that the system is alive and functioning, and everything that happens after that is the runtime phase.

In addition, the initialization of ExternalInterfaces was separated out into ExternalSystem. Code that relies on DLL/CC during system startup should have this system as a prerequisite. Also, prerequisites were added to ensure that on return from snapshot without quitting, subsystem ordering was better preserved, since at that time we can't rely on the distinction between early system initialization and returning from snapshot.

Interval Hashing

Previously, Interval's hash method did not conform to the invariant that equal objects should hash equally. For example,

```
#(1 2 3)  = (1 to: 3)
```

but

```
#(1 2 3) hash ~= (1 to:  3) hash
```

This was resolved by removing the Interval hash implementation, so that it falls back to the implementation for sequenceable collections. This might affect applications which relied on the previous hash, although it seems unlikely. It could be a performance issue for applications which used very large intervals in hashed collections (e.g., as dictionary keys). However, since they gave potentially incorrect answers when used in those ways, the lower performance is probably preferable.

IPSocketAddress printString No Longer Prints HostName

IPSocketAddress had a printString which would attempt to print the hostname. If the hostname could not be found, for example because the machine was disconnected from the network, it would block the VM until the attempt timed out. This has been changed to simply print the numeric IP address. The format has also been slightly simplified. There is a shared variable on IPSocketAddress called PrintHostName which can be set to revert to the old behavior.

Parcel and Package Unload Coordination

In 7.7.1 we have made the Parcel unloader to be more aware of related packages, whereas the Package unloader has for many releases already been fully aware of related Parcels.

If a user has a script that does something like:

```
(Parcel parcelNamed: 'MyParcel') unload
```

or

```
(Parcel parcelNamed: 'MyParcel') remove
```

or

```
(Parcel parcelNamed: 'MyParcel') unloadLogged: <aBoolean>
```

two Transcript message will be seen:

```
Unloading parcel: 'MyParcel'Unloading package: 'MyParcel'
```

Previously, only one Transcript message was seen:

```
Unloading parcel: 'MyParcel'
```

If unloading a Package or Bundle using a script:

```
(Registry packageNamed: 'MyPackageLoadedFromParcel')
unloadFromImage
```

or

```
(Registry packageNamed: 'MyPackageLoadedFromParcel')
unloadLogged: <aBoolean>
```

only the message

```
Unloading package: 'MyPackageLoadedFromParcel'
```

is shown in the Transcript.

Also note that, for several releases it has not been possible to load a Parcel without it creating a related Package (or Bundle).These and other changes to the Parcel system in 7.7.1 have made Parcels and Packages/Bundles more closely integrated.

GUI

Keyboard Handling Path

VisualWorks 7.7 included a change to the way keyboard events were handled. In 7.7.1 these changes were reverted back to something very similar to the 7.6 architecture for dispatching keyboard events.

The keyboard processor will first broadcast the event to all keyboard consumers via the anyfocusKeypressEvent:, which as the abstract method comment says, should respond to the event, only if it is interested in the event in a way that it would want to know about it regardless of whether the widget was focused or not. Shortcuts are an obvious interpretation of this. The menu bar is placed at the logical front of that broadcast.

If none of the anyfocusKeypressEvent: responders return true, then the event is dispatched to the focused widget via handleEvent:.

To activate the debugging facility, set the KeyboardProcessor.AmbiguousAnyfocusKeyboardReactionLogging shared variable to true. When set, the keyboard processor will not stop at the first anyfocusKeypressEvent: that responds true, but enumerate them all regardless, logging second and successive hits. This allows developers to find conflicts in their global shortcut key mappings. Note that text editor shortcut keys would not be included in this by default, since they are only active when the widget is focused.

keyboardProcessor Instance Variables Removed

KeyboardProcessor is no longer referenced directly by Controller or VisualPart objects filling the role of event handlers. When event handler objects need to determine keyboard processor information, they traverse the widget tree to the topComponent to get the current KeyboardProcessor. This reduces errors that occurred when controllers and views acting as event handlers had keyboardProcessor instance variables that were out of sync with the current view tree.

New VisualPart Tree APIs

All VisualParts now support the childrenDo: method. Other children based queries can be derived from this single method. Such as children. Also parent and findParent: have been added. findParent supercedes the findSuperpart: method added in 7.7.

New VisualPart Bounds APIs

All VisualPart objects now respond to the frame method. For the time being, this is stored in the properties of a VisualPart and is only used by Panel and other classes that want to manage layout of visual objects without using LayoutWrappers.

preferredHeight, preferredWidth, and preferredExtent have been added as well. Nominally, these are simply simplification of the preferredBounds API, but may be specialized to be different.

compositionBoundsFor: is now protected so that when view objects find themselves without a parent, they consider their bounds to be the result of Rectangle zero. This is the response of frame as well, when no frame property is found on a VisualPart.

New VisualPart Subclasses

VisualWorks 7.7 included a number of objects in the Browser packages that have been moved over to the core graphics packages. They include:

PixelSpace

Used for layout as a visual whitespace to provide indenting and gaps.

ClickableGraphic

An arbitrary click widget that posts Clicked Announcements when clicked, and supports different images for different mouse states (idle, mouse over, and mouse down).

New Point Transformation APIs

Point now includes methods for creating translated variants along the vertical and horizontal axes. They include downBy:, upBy:, leftBy:, rightBy:.

Window managers more rapidly released after their windows close

A window manager holds a process in which the activities of one or more windows is run.

Prior to 7.7.1, closing the last window of a given window manager did not (usually) shut down this process immediately, because it was hard to guarantee that another window intending to use it was not about to open. (For example, closing the last window might spawn a dialog or other window related to it that would seek to use the same process.) Instead, the general UI activity cycle checked whether window managers should be shut down. On rare occasions, developers would incorporate calls of checkForTerminate in application-specific code to force earlier shut down of a window manager when fast release and garbage-collection of associated objects was a concern.

In 7.7.1, a window manager whose last window has closed performs a more robust deferred check of whether it should shut down. Related to this change, checkForTerminate has been deprecated; instead queueDeferredEmptyCheck calls terminateIfFinished which calls terminateIfEmpty, the method that is functionally equivalent to checkForTerminate.

Application-specific senders of checkForTerminate can probably be discarded if sent from within the closing window's own process. Senders in external code may or may not be droppable, and if not should switch to using the new methods. It is quite safe for code that sent checkForTerminate to go on sending terminateIfEmpty (or a higher method in the calling chain); it is just less likely to be needed.

Few customers will be involved in UI programming at such an intra-framework level. For most, a slight improvement in responsiveness and/or garbage collection will be the only sign of this change.

ApplicationWindow Deprecated/Aliased to ScheduledWindow

ApplicationWindow and ScheduledWindow were collapsed into a single class, ScheduledWindow remaining as the preferred class. ApplicationWindow was added to the VisualWorks library at a time when the framework was evolving and layering the behaviors separated the new extended behavior from the old. As the ApplicationWindow has become mainstream, having two classes leads to confusion for programmers, uncertain where the boundary between the two is, where to add extension methods, etc.

To mitigate backwards compatibility issues, ApplicationWindow still exists as a global shared variable, initialized to be a direct reference to ScheduledWindow. In this way, both ScheduledWindow and ApplicationWindow point to the same behavior. What this means, is that parcels/packages/fileins with extension methods to ApplicationWindow, or subclasses of the same, will continue to load fine.

There are a few exceptional cases to be aware of with this class aliasing scheme. If a client has a like named extension method to both ScheduledWindow and ApplicationWindow, whichever one was loaded second would be the one that showed up in the image. Another is that as these references to ApplicationWindow are loaded in an image, if they are saved out again (i.e. fileout, parcel, or package), then they will now be subclasses of, or extensions to, ScheduledWindow. If these newer versions are loaded into an older image where ApplicationWindow and ScheduledWindow are distinct separate behaviors, then it could lead to unexpected results.

TransientWindow.CachedPopUp and TransientWindow. CachedPostedMenu shared variables Removed

The two TransientWindow class shared variable caches have been removed. They were created at a time when creating windows was deemed to be expensive. They came at the cost of having to manage the various cache invariants that go along with any caching scheme. And have been largely ineffective since the advent of sub menus which guarantee there is often more than one menu active a time. Special methods (maybeRecycle and recycleTransient) for closing these windows but keeping them in the cache were removed as well, simply use closeAndUnschedule instead.

TransientWindow can have a Master

TransientWindows can have an owner window at creation time now, similar to ScheduledWindows. Managing this, keeps some operating systems from thinking the current window has changed focus when TransientWindows are created.

New Tooltips-HoverHelp package Integrated

In VisualWorks 7.7, a new tooltip framework was introduced via the package Tools-HoverHelp. That package has been dissolved, the code all being integrated into the core graphics/interface frameworks. No significant changes were made to its API during the integration. A number of common method overrides were eliminated by doing so.

InputField character limit moved up to TextEditor

Prior to VisualWorks 7.7.1, a character limit could be set for InputField widgets (single line text fields). This behavior has been factored up the class hierarchy so it is now available to other types of TextEditors.

Tools

VisualWorks Project Manager (LaunchPad)

New for 7.7.1 is an application which allows you to manage (create, launch or delete) your VisualWorks development images in a location separate from the VisualWorks install location. VisualWorks is, by default, installed in a location where standard users do not have permission to write files, and creating images there often caused problems. The new LaunchPad application allows you to create projects, consisting of a directory containing a like named VisualWorks image, wherein you can manage your own VisualWorks development independent of the VisualWorks release.

The normal Windows desktop shortcuts, as well as the start items on all platforms, now start up the LaunchPad application. In addition, the installer now exposes a checkbox allowing you to specify placing an applet on the Mac OSX desktop. Like the Windows desktop shortcut, this applet will start up the LaunchPad application.

With the LaunchPad application, you can

- Create and launch a new project (with the [+] button)
- Launch an existing project (with its arrow button)

- Remove an existing project (with its [-] button)
- Change the VisualWorks Projects root directory (with the folder button)

The default VisualWorks Projects root directory is a subdirectory of the standard My Documents location on Windows. This is typically

```
C:\Documents and Settings\<username>\My Documents\VisualWorks
Projects
```

or

```
C:\Documents and Settings\<username>\Documents\VisualWorks
Projects
```

or

```
C:\Users\<username>\Documents\VisualWorks Projects
```

On Mac OSX and Linux/Unix platforms this is a subdirectory of the standard $HOME location, typically

```
/Users/<username>/VisualWorks Projects
```

or

```
<root>/home/<username>/VisualWorks Projects
```

The VisualWorks Projects root directory is persisted in the environment variable, VWPROJECTS. This is managed automatically by the LaunchPad application on Windows (through the Windows Registry) and on Mac OSX (in the VM's .plist file). On other Linux/ Unix platforms you will need to manage setting this environment variable in your shell scripts in exactly the same way you currently manage setting the $VISUALWORKS environment variable.

Please note that the LaunchPad application is a 32-bit only application for VisualWorks 7.7.1. If you are using a 64-bit platform and running the 64-bit installer, we install both the 32-bit and 64-bit VMs in order to accommodate the LaunchPad. If you choose not to install the 32-bit VM on a 64-bit platform, you will not be able to use the LaunchPad application.

Comparison Tool

For package to package comparisons, VisualWorks 7.7.1 includes a new compare tool. It is architected from the ground up to optimize the process of comparing code.This revision has some history. The original compare tool written for 7.7.1 is also still in the system. It is not the same as the one found in 7.7, but follows a similar approach, using lots of list widgets with menus, and being heavy on “change management" rather than "code comparison." The Store settings page has a "Use New Compare Tool" option which can be disabled if you'd rather use the old tool. Also, if the SHIFT key is held down when opening a comparison browser, that particular open, will open the original 7.7.1 tool, rather than the latest one.Under the hood, the new compare tool is all about blueprints and the Meyer's diff algorithm, which is now implemented in VisualWorks. The base model is actually completely divorced from Store, but can draw data from Store objects as one of its sources of blueprints.An example of using the same tools to do something completely independent of Store, can be seen by evaluating the expression

```
ClassBlueprintComparisonView compareTwoClasses
```

Try it on `Float` and `Double` for fun.Currently the only integrated visual tool access to it is with Store components though, from the various compare menu options in the IDE.The compare tool has two basic view elements. A "header view" at the top, and a "main view" below that. The header view has two modes, summary and detailed. In summary mode, it only shows one line, which summarizes how many packages were changed between the two. Expanded to details, it shows each package that was changed (or added or removed). There are menus on the packages shown in the header for browsing, listing versions, loading, and filing out differences. The +/- icon at the bottom of the header toggles between summary and details view.By default, the compare tool will always attempt to put "newer" code on the right. Image code is correlated with the timestamps of the package it was loaded from. So even if you've made changes to a package that are "newer" than the version you are comparing to, the original timestamp of the package still may predate the one being compared with. In some cases though, when comparing against image code, we cannot determine a useful time reliably (e.g., code loaded from parcel). The convention when this happens is to always assume the image code is newer (and goes on the right) if we cannot find a date for it. There is a side swap button that spans top to bottom between the two header sets, in the header view. Clicking it will flip the sides of the comparison.Which changes are shown in the main comparison area, can be filtered by disabling lines in the header view. They are by default, all checked green. Toggling this, removes those changes from the scope of consideration. Toggling one of these with the SHIFT key held, enables that item and disables all others, thus being a quick way of selecting just on of the packages and what changes are in it.On purpose, while the packages are shown in the header, they are not shown in the main comparison area as a structuring element. Much code development involves moving things between packages, and one does not want to have to navigate between two different package selections to see that it is simply moved. While they may not be there as a structuring element, when a packaging change has happened that is shown. The ... icon at the tail of most items in the view though, has a popup window that displays package and other information for that item.The comparison area is essentially a tree view with expandable/collapsable elements. Some views show summaries of what you will see when you expand them.An object that is entirely new will show up in green with the + icon annotations. One that is entirely removed is shown as red with x icon annotations.Rather than showing class/namespace/share definition messages as changed, the different parts of these objects are compared independently.Another thing it will do, is try to guess at methods that have been renamed. When it finds an added and removed method, which no other changes, an elliptical line is drawn between added and removed methods, indicating it was likely simply a rename.There are lot of little icons in the comparison tool, and a number of keyboard modifiers. The help button opens an overview of the compare tool, which includes an icon reference and a keyboard shortcut reference.

SUnit

The following changes relate to SUnit version 4.0 of the Camp Smalltalk SUnit project, included in VW7.7.1. SUnit 4.0 preserves full backward compatibility: there are no API changes. All changes have also been already released in some other Smalltalk dialects and are being ported to the remainder, preserving the cross-platform behavior of SUnit.

1 A number of users of SUnit have asked that it be made easier to plugin special behavior. In SUnit 4.0, the TestResult handler dispatches to the exception raised by the test to determine how to react to a TestFailure or Error. The exception in turn invokes a method on the TestResult to cause this reaction. Thus, users can add specialised behavior by creating exception subclasses or by overridding the methods they invoke in specialised TestResults. The skip pattern (see classes TestSkip and SkipResource in the add-on parcel SUnitResourcePatterns) is an example of using this.

2 Previously, the setUp call of TestResource was put inside its initialize call. This meant that a resource that failed in setUp never returned

its instance from new, and so could fail to be torn down through lack of a reference to it.

- In SUnit 3.1 and earlier this, though often encountered as a problem, was usually accompanied by such evident aborting of the whole run that problems in subsequent runs were often guessed to be related to the failure and some hand-hacking could then regain a clean resource state. In SUnit 3.2 and after, resources are more robust in general, so a failure to call tearDown is less strongly signalled.
- TestResources were unlike TestCases, which ensures that if a test's setUp is entered, its tearDown will be called. (This sometimes required ifNotNil: guards in tearDown statements so that earlier failures that left instVars unexpectedly nil did not then cause additional trivial errors in tearDown. User experience is that the need for these guards is easy to spot and fix.)

In SUnit 4.0, TestResource tearDown will be called if setUp is entered. Users may wish to add ifNotNil: guards to TestResource tearDown statements. We believe the overall gain in robustness makes this change well worth doing. Test resources invoked via the framework API are set up as before; users who always use resources by adding them to class-side resources methods, as recommended, should see no changes. If a user-created script sends new to a TestResource class, bypassing the framework API, it may need rewriting to call setUp explicitly. (Generally, any user whose test programming style makes explicit calls to test resources without also referencing them in class-side resources methods should read the testing chapter in the tools documentation and make themselves familiar with the changes to class TestResource.)

3 SUnit 3.2, provided in VW7.7, improved the ordering of test resource setUp and tearDown but did not handle potential edge cases when several resources used the same resource. SUnit 3.3 fixed these edge cases.

Resources are now torn down order-reliably by the method TestResource class>>resetResources:. Any user scripts that invoke resource-using tests in ways that bypass the framework should use this method.

4 SUnit's approach to getting test selectors was unperformant, getting all the many selectors of Object and then discarding them,

which could matter in certain specific performance-testing scenarios. This code has been improved.

5 New method TestCase class>>lookupHierarchyRoot supplements TestCase class>>shouldInheritSelectors in controlling inheritance.

The following changes are VW-specific.

6 RBSUnitExtensions handles inherited test selectors and extended test selectors more rationally; see the Testing chapter's discussion of the RBSUnitExtensions utility for details.

7 SUnit runs its tests in order by default. RBSUnitExtensions does the same but in VW7.7 and earlier, its order could differ from that of basic SUnit and not be what the user would expect. In VW7.7.1, the order is the class order, with alphabetic order of selectors within classes, that the UI implies.

8 Clicking **List Defects** no longer opens a dialog (the VW7.7 behavior) but a browser on the failed and erring tests. A test can be selected and debugged in this browser exactly as in the list provided by the dialog, but it can also be rerun, edited, etc. (A variant implementation that keeps the dialog, adding a button to it to spawn the browser, is in the Cincom open repository.)

Method Icons

VisualWorks 7.7.1 adds some additional method icons, such as an icon for deprecated methods, one for methods that participate in the construction of menus, and a general one for methods that have some sort of <method tag> in their source.

The subimplementor and superimplementor method indication icons, have been changed to be small triangle arrows, which can be used to overlay other method icons. In this way, we now can show both that a method is, for example, a menu method and has a superimplementor at the same time.

RefactoringBrowser Pundle Environments

The Smalltalk browser can now be opened on a subset of packages found in the system. Prior to VisualWorks 7.7.1, it was either the whole system (all packages) are just one package. There is now a **Spawn Selected** menu item in the packages list which can be used to spawn the current package. Unlike previous version, this new browser, even though it has only one package shown in it, will include the package list, so that package operations can be done on that package. But in addition, the developer may select more than one package and have all of those browsed at once.

There is also a **Spawn…** submenu in the package list. The submenu has a number of different computed sets of packages that can be spawned. One can spawn all the packages with unpublished changes in the system. Or all of the code marked with Cincom copyrights, or vice versa, which gives a quick way of separating Cincom code from customer code.

Adding your own spawn submenu item is extensible. See the various spawn* methods found in BrowserNavigator menus method category, for examples of how to add your own.

RefactoringBrowser Package List Presentation

Moving to subsets of packages shown in a browser, led to some interesting discoveries about our traditional sort order show in the browser. The classic mode of sorting bundles to top, and then sorting bundle sub contents by load order, does not scale very well when potentially only showing some of the packages. To simplify things, we now sort alphabetically by default, bundles to the top. There is a **sort** submenu in the package list menu that allows this to be toggled between the traditional load order and the newer alphabetic sort for a particular browser. It may also be set globally in the system settings tool.

The other sort option that shows up in the **sort** submenu, is the ability to choose whether to sort packages separate from bundles, placing the bundles at top, or whether to intersperse them all alphabetically. The default way, is more like what users might see with MicroSoft Windows file explorer tools. And the latter is more like what one finds when using Mac OS X. This may be set per instance, or globally as well.

RefactoringBrowser Bundle Load Order Changes

There is now a first class **Bundle Structure** tab in the lower half of the browser, which activates when a bundle is selected in the package list. This tool shows the subcontents of the bundle, in their load order (regardless of the package list sort order). There are simple widgets for adding new ones, and removing existing ones. Order can be changed by drag and drop.

Bundle load order is now validated real time as well. Bundles with invalid load orders, will show a warning icon on them, and the **Bundle Structure** tab. When there are load order issues, a count of errors shows at the bottom of the **Bundle Structure** tab. If clicked on, it will open a more detailed window of the load order dependencies that are in violation. Showing this warning in the browser, can be toggled off/on right next to the warning label at the bottom. This may be useful for bundles such as **Base VisualWorks** which have known load order issues, which don't really turn out to be issues.

Timestamps Displayed in the Local Time Zone

Store uses UTC timestamps, based on the database time. But these can be confusing in display, so the tools now convert the timestamps to the local time zone for display purposes.

Database

Server Connection Notifications

Information from the server at connect time is now available as a Notification, which can be ignored, or caught as a resumable exception. In this way, for example, a notice that the user needs to his or her password, can now be caught and presented to the user. For example, this workspace code shows how to handle password expiration gracefully:

```
[   conn := OracleConnection new.
    conn environment: 'OracleDB';
        username: 'user';
        password: 'password';
        connect.
    sess := conn getSession.
    sess prepare: 'select * from dual'.
    sess execute.
    ansStrm := sess answer.
    res := ansStrm upToEnd. ] on:
        Exceptiondo: [:exception | Dialog warn: exception errorString.
    (exception isKindOf: Notification)
        ifTrue: [ exception pass ]
        ifFalse: [ exception return: nil ].].
```

Oracle Buffer reuse

We have introduced a feature to allow OracleBuffers to be reused. If the developer knows in advance that a second query will need exactly the same buffers as the preceding query, then the following message can be sent to the session object, which serves to bypass all column size and type checking:

```
aSession reuseColumnBuffers: true.
```

This remains in effect for the session until it is turned off using:

```
aSession reuseColumnBuffers: false.
```

The default setting is false, so legacy code should be unaffected. To use this feature, one must be certain that the query will require the same buffer types and sizes as the previous query; otherwise, C pointer errors are likely, and these can lead to data corruption and image crashes. The feature is intended for experts who are optimizing well understood code.

Consistent use of UTF-8 Encoding for Postgresql Database

In previous versions, some facilities would use UTF-8 to communicate with a Postgresql database, but others would use the #default encoding for the platform. This potentially resulted in incorrect interpretation of non-ascii characters in, for example, package names. The facilities now consistently use UTF-8 throughout.

OracleEXDI Timeout Error

When using OracleEXDI on MacOXS, Linux and Windows 7, connetion may fail with the following Oracle error:

```
01804, "failure to initialize timezone information"// *Cause: The timezone
information file was not properly read.// *Action: Please contact Oracle
Customer Support.
```

In VisualWorks 7.7.1, we use a Notification to catch this error. Users can choose either to handle or ignore it (by default, it is ignored). The following is an example of catching and displaying the error:

```
[   conn := OracleConnection new.conn username: 'username';password:
'password';environment: 'oracleDB';connect ] on: Exception do: [ :ex|
Dialog warn: ex errorString.(ex isKindOf: Notification) ifTrue: [ ex pass ]
ifFalse: [ ex return: nil]]."Try to execute a query."sess := conn
getSession.sess prepare: 'select count(*) from sys.all_tables'.sess
execute.ans := sess answer.res := ans upToEnd.
```

Ad Hoc SQL Tool Enhancements

The Ad Hoc SQL Tool has been upgraded slightly in two ways.

First, the displayed column widths are now fully adjustable by the user. In addition, there is a new menu setting, "Display/Fast Truncation." If checked, it initially displays the columns in the usual widths. Unchecked, the tool calculates optimum display widths, based on the largest retrieved item in each column. This option can take noticably longer for large data sets.

The second change is the addition of a simple spin button to recall previously entered queries. This option saves typing, and serves as a history of past queries. The history is lost as soon as the tool is closed.

ExternalInterface for 64-bit Oracle

ExternalInterface definitions for 64-bit Oracle libraries (Linux and Solaris) were previously defined with specific version suffix (**`libclntsh.so.10.1`**). Oracle EXDI isn't usually as sensitive to Oracle version changes, so we aligned those with the 32-bit interface definitions and drop the version suffix.

For the Oracle server product, you have to execute

```
$(ORACLE_HOME)/bin/genclntsh
```

in order to get the version-less library names installed.

Alternatively, the following configuration can be used with recent Linux versions, allowing concurrent use of both 32-bit and 64-bit versions. Let's assume both 32- and 64-bit Oracle client installation. The default location (using the rpms from technet) are following directories:

```
libclntsh.so.11.1 (libc6,x86-64)
   => /usr/lib/oracle/11.1/client64/lib/libclntsh.so.11.1
 libclntsh.so.11.1 (libc6) => /usr/lib/oracle/11.2/client/lib/
libclntsh.so.11.1
```

Add generic **`libclntsh.so`** links to **`/usr/local/lib`**(64) directories:

```
/usr/local/lib/libclntsh.so
   -> /usr/lib/oracle/11.2/client/lib/libclntsh.so.11.1
 /usr/local/lib64/libclntsh.so
   -> /usr/lib/oracle/11.1/client64/lib/libclntsh.so.11.1
```

Add the following **`ld.so.config`** files:

```
/etc/ld.so.conf.d/oracle.conf:/usr/lib/oracle/11.2/client/lib/usr/lib/oracle/
11.1/client64/lib/etc/ld.so.conf.d/local.conf:/usr/local/lib/usr/local/lib64
```

Run **`ldconfig`** as root, so that the library cache gets rebuilt. You can verify that the dynamic linker can see the libraries with

```
ldconfig -p | grep libclntsh
```

With this setup the external interface definition does not need to be modified for specific library version. The only requirement is that **`libclntsh.so`** points to a functioning installation.

Store

Prevent duplicate version strings

Store now adds a unique constraint to the BUNDLE and PACKAGE tables, to make sure that for any specific Package or Bundle, the same version string is not duplicated.

Store also adds two indexes, one each to the METHOD and METHODS table, to speed up quearies to the database.

To add these new behaviors, have the database administrator run:

```
Store.DbRegistry update771
```

Note: If your database already has duplicate name+version for bundles or packages, the adding of the above unique constraints will fail for the table where duplicates exist.

Creating DB on SQLServer 7.x not supported

Beginning with this release, VisualWorks no longer supports CREATING a Store database on SQLServer 7.x. We continue to support existing Store SQL Server 7.x databases.

New Store Browsers

As part of our ongoing effort to completely obsolete the OldBrowsers package, the Store Browsers now use the standard System Browser (Refactoring Browser) UI. These browsers present a read-only view of source published to a Store repository and can be used for browsing specific versions of bundles and/or packages, with their contained classes, methods and shared variables. Because the Store

Browsers work with static representations of method source as opposed to operating in a "live" code environment, you cannot perform some of the operations common to the System Browser, such as searching senders/receivers of selectors in the method source, or any of the refactoring actions.

Some Store Browser classes inherit from their Refactoring Browser counterparts, but since the standard System Browser is perhaps the most extended tool in the VisualWorks IDE, care has been taken to isolate the Store Browsers from the potential impact of Refactoring Browser extensions. The primary browser classes, RefactoringBrowser, BrowserEnvironment, BrowserNavigator, NavigatorState, and CodeModel have all been refactored to provide a superclass so that its Store Browser couterpart is a sibling and not a subclass of the System Browser class. In addition, toolbar and menu items for all the Store Browser classes are explicitly implemented in that class, and any menu augmentation also terminates with that class.

As with all of Store, the database access for the Store Browsers now uses the Glorp database framework. This means that the objects being browsed are pulled into the image with Glorp proxies and the browser content is database session dependent. Therefore, whenever you close a repository connection you will be warned if there are any remaining open Store Browser or Store version list windows. If you choose to continue with your disconnect request, these windows will be closed. Likewise if you save an image with one or more of these tools open, when the image is closed and reopened it will not re-open any of these UIs.

Warning when publishing from an older version

When publishing, if someone else has also published a version descended from the one your version is based on, you will get a warning dialog. Previously the only indicator was that the version name would be branched, and only if the person had chosen the default version name.

Merge Tool, Version Browsers Now Based On StoreGlorp

The Store Merge Tool has had its internals entirely rewritten and is now based on the StoreGlorp database objects. The user interface remains mostly the same, but extensions or modifications of the tool may need to be changed. The same is true of the various version list browsers, the **Browse Versions** option in Store menus for most entities. Browsing packages in the database is also changed to be based on the Refactoring Browser, as noted elsewhere.

Better Separation of UI and Domain Objects

The use of StoreGlorp in general provides a better separation of the Store domain objects from the tools that use them. In this release we have also removed many uses of hard-coded dialogs within the Store code, instead raising exceptions whose default action is to open the appropriate dialog. Automated usage of Store APIs can catch these exceptions (under `StoreError`) and react appropriately rather than needing user intervention.

ENVY Bridge Removed

The ENVY Bridge add-in always provided only minimal support for porting to Store, and has since become thoroughly out-of-date. Due to the reduced need for such ports over the years, the supporting parcels and documentation have been removed from the product.

For any customers still needing to port from ENVY to Store, contact customer support for help and consulting options.

Store Garbage Collection

Starting with 7.7.1, the Store Garbage Collector has a new UI and engine. The new Garbage Collector is now Glorp based, and no longer uses the older Store DB Objects.

This new engine works slightly differently than the old one. The new engine is a two step mechanism. It first marks targeted packages and bundles in the database with a blessing of Marked For Deletion (blessing level -54), and then executes a garbage collection on all packages and bundles marked with that blessing level.

The new UI allows the user to mark items for deletion, without actually having to start the garbage collection engine. This allows the user to mark and unmark items for deletion and later fire off the actual garbage collection.

The user can either use the new GarbageCollection tool to mark and unmark bundles or packages for deletion. To only mark a package or bundle for deletion you also can use the Set Blessing Level... menu options in standard tools.

Once packages or bundles have been marked for deletion, exeuting:

```
GarbageCollector collectGarbage
```

starts the garbage collector.

The StoreForGlorpReplicator has been updated to pay attention to packages and bundles that are marked for deletion, and will NOT replicate any bundle or package marked for deletion.

In this way, if a replicator is set to cross replicate two repositories, marking items for deletion in one or both, will make sure that the replicator won't back fill removed packages or bundles.

WebServices

Updates for WebServices 2.0

Implemented features

This release introduces WSDL 2.0 support implementing the W3C WSDL 2.0 specification with Soap 1.1 and 1.2 binding extensions. It allows parsing WSDL 2.0 documents into WSDL 2.0 components, and validating their syntax and semantics. Soap 1.2 messages can be created, sent and received according to valid WSDL 2.0 descriptions.

The WSDL 2.0 specification was published as a W3C Recommendation on June 26, 2007.

http://www.w3.org/TR/wsdl20/

The SOAP 1.2 specification was published as a W3C Recommendation on April 27, 2007.

http://www.w3.org/TR/soap12-part1/

This release includes updated WebServices framework supporting both WSDL 1.1 and 2.0 versions, and Soap 1.1 and 1.2 versions. The set of supported Wsdl 1.1 features remains the same as it was for 7.6 release. This release includes the following new features:

- Parsing WSDL 2.0 into component models and validating the schema.
- Parsing SOAP 1.2 binding extension.
- Creating, sending and receiving SOAP 1.2 messages by WsdlClient and Opentalk-Soap client.
- Opentalk server supports both WSDL 1.1 and 2.0 protocols.
- SOAP 1.2 encoding for unique identifiers (id/ref).
- SOAP 1.2 RPC and IRI styles.

- SOAP 1.2 action feature.
- SOAP 1.2 HTTP binding (GET & POST).
- SOAP 1.2 Headers.
- SOAP 1.2 Fault model.
- SOAP 1.2 MEP: request-response and soap-response

Unimplemented functionality

There is no Web services tool support for Wsdl 2.0 yet. Here is the detail list of features still to be implemented.

- Generating WSDL 2.0 schema
- Generating domain, client and server classes from a WSDL 2.0 schema
- Using the id attribute to identify inline schemas
- SOAP 1.2 arraySize attribute.
- Interface operation multipart style.
- WSDL 2.0 HTTP binding extension.
- SOAP 1.2 nodeType attribute
- WSDL 2.0 the safety extension interface operation property (wsdlx:safe="xs:boolean")

Changes in the WebServices framework

Soap request return result

In accordance with the Basic Interoperability Profile, request result now always corresponds to the element type as defined by the schema types section. This means that result of operations defined in WSDL 1.1 using document style is not wrapped in a collection anymore.

To ease the transition, a global backward compatibity option was added (SoapWsdl11OperationBinding class>>#wrapDocumentLiteralResult) which restores the original behavior.

Packaging

- The SOAP parcel was merged into the WSDL parcel. Prerequisite references to SOAP should be changed to WSDL.
- WsdlClient>>createScript and WSOpentalkClient>>createScript methods were moved to WsdlTool parcel.

Soap Headers

SoapHeader is not a Struct anymore because header entries can repeat. Instead it holds an ordered collection of SoapHeaderEntry instances. Based on this there are some changes in the SoapHeader API:

SoapHeader>>headerAt:put:

Adds a new header entry to the end of the collection

SoapHeader>>headerAt:ifAbsentPut:

If the header entry is absent, adds the new header entry to the end of the collection

SoapHeader>>remove:

Detects the first header entry that match the parameter and removes it from the collection. If there is no match the method returns nil

SoapHeader>>removeKey:ifAbsent:

Detects the first header entry that match the parameter and removes it from the collection. If there is no match the absent block is called.

SoapHeader>>add:

Adds a header entry or collection of header entries to the collection

Consequently the SOAP header entries are not registered in the process environment individually. Instead the header as a whole is registered under the key #SoapHeader.

```
soapHeader :=(ProcessEnvironment current
   at: #SoapHeader ifAbsent: [nil].
headerEntry :=  soapHeader headerAt: 'AuthenticationToken'
   ifAbsent: [nil].
```

Also, the first parameter in WSHeaderEntryProcessor processOutputHeader:reply:transport: is now a SoapHeaderEntry instead of an Association. See the updated example in WebServicesDemo:

```
HdPrConfirmation>>processOutputHeader: aHeaderEntry reply:
aSOAPReply transport: aTransport

    | confirmation |
    ((confirmation := aHeaderEntry value)
        isKindOf: WebServices.Confirmation)
        ifFalse: [ self error: 'Wrong confirmation value '  ].
    ...
```

Framework refactoring

- Some classes were refactored to allow consistent support for Soap 1.1 and 1.2 bindings.
    - SoapFault class is now a superclass of Soap11Fault and Soap12Fault hierarchy.
    - SoapHeaderEntry is now a superclass of Soap11HeaderEntry and Soap12HeaderEntry.
- There is new class hierarchy that is responsible for marshaling/unmarshaling SoapRequest and SoapResponse.

```
SoapOperationBinding
    SoapWsdl11OperationBinding
    SoapWsdl20OperationBinding
```

- There is new SoapTransportBinding class that is responsible for marshaling SoapFault objects.

Unique identifiers

XMLObjectMarshalingManager has new class instance variables: refNode and idNode allowing to set-up the tags for unique identifier elements. The default set-up are unqualified tags: "id" and "href" for backward compatibility.

SoapMarshalingManager sets these variables based on Soap binding version: for Soap 1.1 it uses "id" and "href" and for Soap 1.2 it uses "enc:id" and "enc:ref" (enc:xsd="http://www.w3.org/2003/05/soap-encoding").

Consequently all shared variables of XMLObjectMarshalingManager were converted to class instance variables.

IRI encoding

Soap 1.2 binding supports sending SoapRequests using GET method where the request parameters (which must be simple) are encoded directly into the URL. GET method doesn't allow to carry a body so it cannot transport XML. This style can also be useful for creating a REST-like facade on top of a standard WebService.

As an example let's consider a service with the following fragments of WSDL definition:

- the service address is defined as

```
address = "http://ws.example.com/service1"
```

- the interface operation defined as

```
<operation name="getTemperature"
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri">
      <input messageLabel="In" element="getTemperatureIn" />
   </operation>
```

- the operation binding defined as

```
<operation
   ref="tns:getTemperature"
   wsoap:mep=
      "http://www.w3.org/2003/05/soap/mep/soap-response/"
   whttp:location="temperature/{town}"
   whttp:queryParameterSeparator="&"/>
```

- the schema type getTemperature defined as

```
<element name=getTemperatureIn>
      <complexType>
         <sequence>
            <element name="town" type="xsd:string"/>
            <element name="date" type="xsd:date"/>
            <element name="unit" type="xsd:string"/>
         </sequence>
      </complexType>
   </element>
```

With the above definition, invoking the operation getTemperature with parameters

```
#('Paris', 2007/06/26 'C')
```

will result in an HTTP GET with the following URL:

```
http://ws.example.com/service1/temperature/Paris?date=2007-06-
26&unit=C
```

Soap Action

Wsdl 1.2 and 2.0 handle Soap action differently. In the SOAP 1.2 HTTP binding, the SOAPAction HTTP header defined in SOAP 1.1 has been removed. SOAP 1.2 utilizes the "application/soap+xml" media type to transmit XML serializations of SOAP messages. The media type specifies an optional action parameter, which can be used to optimize dispatch or routing.

```
Content-type: application/soap+xml;action=
   "http://greath.example.com/opCheckAvailability"
```

Consequently we changed how we use the Soap action string to find an operation.

The old defaultSoapActionBlockValue block extracted only the operation name from the string and used it to find an operation. For example, consider

```
soapAction = 'http://webservices.cincomsmalltalk.com/
UnitConverter#addUnit'.
```

The old implementation extracted 'addUnit' and found an operation with the name matching 'addUnit'. The new implementation compares the whole string 'http://webservices.cincomsmalltalk.com/UnitConverter#addUnit' with an operation Soap action to find matching operation. This change relaxes the constraints imposed on the soapAction structure.

Internationalization

National Currency Symbols

This release adds National Currency Symbols to currency formatting in CLDR-derived locales.

Only region-specific locales (those with both a language and a territory component in their names, such as “en_US”, but not “en”) have currency symbols added because only regions have currencies; culturally-neutral language-only locales do not.

If there is no currency symbol for a designated currency, the ISO4217 abbreviation is used, as specified in the CLDR.

If no symbol nor ISO4217 abbreviation appears for currency formatting in a region-specific locale, it is because for one reason or another there is no valid currency for that locale at the time of the

CLDR version from which these initialization methods are derived. For example, some locales are associated with defunct states, which logically have no currency.

If there is no optional format for negative currency amounts for a given locale, it is because the source format in the CLDR does not contain one. This varies by locale.

Special Character Handling

Special characters are no longer stripped from the CLDR during processing. For our purposes, special characters consist of the non-breaking space (x00A0), the currency symbol (x00A4), left to right embedding character (x202A), and the pop directional formatting character (x202C).

The currency symbol referred to here is not the same as the National Currency Symbol, but a sort of concave-sided box that is a placeholder for inserting the National Currency Symbol.

These characters will become much more important when we support composing with Pango or another API.

This has implications for spacing with regard to currency symbols in print formats. In some cases, as in the USA format, there is no space between the currency symbol and the numbers. In other formats there may be a space, and if so, it is usually a non-breaking space, so as to keep the currency symbol together with the currency digit information when displaying or printing.

In some cases such a print policy format does not have a valid currency symbol, which would leave the format beginning with an unnecessary non-breaking space. In these cases, the non-breaking space is trimmed. The disambiguation of the currency symbol by various locales is not standardized in the CLDR. For example the US locale uses “US$” while some other locales use simply “$” (or another symbol) for their own currency without further disambiguation. In all cases we have transferred the currency symbol as it appears in the CLDR without making editorial adjustments.

Choice Patterns

The CLDR allows for “choice patterns,” which allow a runtime decision as to what form of the currency symbol is used, based upon the magnitude of the currency amount being displayed. This appears to be limited so far to locales in India, and the variations consist of different pluralities of the Rupee designation.

Since we do not support runtime composing at this time, we have chosen the most reasonable default based upon CLDR guidelines. This is something that could be done differently in the future.

Opentalk

Load Balancer Obsolete

The Opentalk based load balancer is now obsolete and will be removed from the product in following release.

Net Clients

Mail Xfer Encoding

In this release we started applying content transfer encoding to all mail message parts. The mail messages part with content type "text" will be encoded as #quoted-printable and all other types as #base64.

The old implementation added base64 transfer encoding only to parts with binary contents.

If you don’t want to apply the transfer encoding to a message, you can do either of the following:

- encode the part and add the header field: Content-transfer-encoding: someEncoding, or
- turn off the writing by sending `applyTransferEncoding:`

```
message := MailMessage newTextPlain.message writingOptions
applyTransferEncoding: false
```

Glorp

Login Objects Forget Passwords After Use

Glorp `Login` has new instance variable, `secure`, initialized to `true`. When it is true, the Login password is nilled immediately after it is accessed, normally upon login. A secure Login cannot be reused without the password being again provided. This improves security and allows compliance with policies which forbid applications to store unencrypted passwords in memory for longer than necessary.

This change may causes issues in existing applications where logins are reused. If the security concerns do not apply, developers should ensure the appropriate logins set secure: false. Alternatively, change Login to initialize secure to be false if that is acceptable in your environment.

SQLite Support

Limited SQLite Active Record Support has now been included with Glorp.Because SQLite does not implement the standard information_schema views, the full suite of queries against all kinds of database objects is not possible. Conformant DBMS allow for a wide variety of MetadataDescriptor queries, returning, for example, all foreign key objects whose target table is named 'mytable'. Unfortunately, with SQLite, the only MetadataDescriptor query possible is one which returns table objects.That said, the system does return complete table objects and all their related objects (a complete, closed system of related tables, columns, keys, constraints, etc.). So, for example, it is a simple matter to query all tables, which returns the entire database information schema.

CairoGraphics Preview

VisualWorks 7.7 shipped with preview parcels to bind the CairoGraphics 2D drawing library with Smalltalk. An updated version for 7.7.1 has been created, which makes adjustments for changes in the way drawing surfaces are managed in the Mac OSX VM. There are no updates to the libraries (there are slightly newer versions of them, but resources didn't allow us time to build new ones).

One restriction that came up in the last minute of the 7.7 release, was that one could not reliably draw on OSX Pixmap surfaces. This is no longer the case.

Another issue that has arisen with the provided library, is that it doesn't play well with Pango pre built binaries downloaded from other sources. We hope to provide an updated stack of Pango and CairoGraphics parcels and libraries (for OSX and Windows)some time shortly after the release.

Documentation

This section provides a summary of the main documentation changes.

Note that we changed the documentation source format in 7.7. While we have attempted not to lose any content or formatting, some errors have nearly certainly occurred. Please notify us of serious lapses.

Basic Libraries Guide

- Update Date creation description
- Add creating number from string

Tool Guide

- Update for changes to System Browser
- Update Unit Testing chapter

Application Developer's Guide

- Add Smalltalk Archives as a source format.
- Update System Browser information
- Update Install as Service instructions
- Add -nologo switch
- Update binary selectors description to ANSI spec

COM Connect Guide

- Update default specification policy

Database Application Developer's Guide

- Add information about Oracle password support
- Add OracleEXDI #reuseColumnBuffers: information
- Add ODBC THAPI examples

DLL and C Connect Guide

- Remove SGI support
- Note limitation on FixedSpace

- Add platform information for Solaris
- Add/correct C API mapping example (chap 3)

DotNETConnect User's Guide

Miscellaneous updates and corrections

DST Application Developer's Guide

No changes

GUI Developer's Guide

- Correct Painter load instructions
- Remove Notebook widget (obsolete)

Internationalization Guide

- Major revision, updating for CLDR locale support

Internet Client Developer's Guide

No changes

Opentalk Communication Layer Developer's Guide

- Remove load balancing feature (unsupported)

Plugin Developer's Guide

No changes

Security Guide

- Expand/update X.509 support

Source Code Management Guide

- Update Merge Tool description

Walk Through

No changes

Web Application Developer's Guide

No changes

Web GUI Developer’s Guide

No changes

Web Server Configuration Guide

No changes

Web Service Developer’s Guide

Minor updates