Cincom

VisualWorks®

Version 7 Release Notes

P46-0106-05

Part Number: P46-0106-05

Software Release 7

This document is subject to change without notice.

Cincom Systems, Inc.

55 Merchant Street

Cincinnati, Ohio 45246

Phone: (513) 612-2300

Fax: (513) 612-2000

World Wide Web: http://www.cincom.com

Contents

Chapter 1 **Introduction to VisualWorks 7** 7

Product Support Status 7
ARs Resolved in this Release 7
Items Of Special Note 8
- Store Database Format Changed 8
- VM Compatibility 8
- New Toolset 8
- Professional Debug Package 8
- Win95 support discontinued 9
- Pollock 9
- MacOS version requirements 9
- File name lengths on Mac 9
- HP-UX 10.2 VM availability 10

Chapter 2 **VW 7 New and Enhanced Features** 11

Base system 11
- New Collection methods 11
  - Dictionary 11
  - SequenceableCollection 11
  - Collection 12
- New Object methods 12
- Instance Immutability 13
- New Weakness Facilities 13
- Ephemerons 14
- TimeZone and Time 14
- External Interface exception handling 15

GUI Development 15
- UIPainter is TriggerEvent Driven 16
- ValueModel support for TriggerEvents 16
- Dataset 16
- TreeView 16

Window placement 17
Graphics 17
Tools 17
Parcel Manager 17
File browser 18
Browsers 18
Workspace 19
WebService enhancements 19
Secure Socket Layer (SSL) 19
Database 20
Threaded ODBC EXDI 20
Oracle Unix/Linux Interface 20
Oracle 8/9 support 20
XML enhancements 21
VisualWave/Web ToolKit 21
Startup 21
Configuration 22
Security 22
Logging 22
Debugging 23
Demos 23
Documentation 24
Known limitations 24
Tiny HTTP Configuration 24
Operating System Limitations 24
Loading Wave Palette Icons 25
Case Sensitivity in Web Toolkit Pages 25
Buffering 25
Authorization with CGI Gateway 25
Startup Directory and Configuration Files 25
Store 26
Expanded documentation 26
New Connection dialog 26
Prerequisite version control 26
Binary load enhancements 26
Prerequisite specification 27
Performance enhancements 27
Bundle content management 27
Merge tool and bundles 27
Database table alias support 27
Known Limitations 27
Bundle and override usage 27
StoreForSQLServer incompatibility with ODBC 28

Opentalk 28
Exception forwarding 28
Catching Broker Errors 28
Opentalk process priorities 29
Resumable timeout exceptions 30
DST 30
Demo parcel renamed 30
Installation in the presence of Store 30
Virtual Machine 30
Backward compatibility 30
New debugging engines 30
Extensions for Base Support 31
Packaging as an Executable 31
Documentation 31
Comprehensive document index 31
PDF renaming 31
Browser documentation 32
Store documentation 32
Opentalk 32
XML framework 32
WebServices 32
NetClients 32
Events 32
VisualWave/Web Toolkit 33
Help 33

Chapter 3 Deprecated Features 34

Base System 34
isNotEmpty 34
Database 34
Oracle 7 EXDI 34
DST 34
GUI 35
Tools 35

Chapter 4 Preview Components 36

New GUI Framework (Pollock) 36
High Level Goals 37
Pollock 37
Pollock Requirements 38
The New Metaphor: Panes with frames, agents, and artists 39

Other notes of interest 40
So, What Now? 41
Opentalk SNMP 42
Usage 42
Initial Configuration 42
Broker or Engine Creation and Configuration 42
Engine Use 43
Entity Configuration 45
MIBs 45
Limitations 45
Port 161 and the AGENTX MIB 45
Opentalk 46
SocratesEXDI and SocratesThapiEXDI 46
Installation 46
SocratesXML 1.2.0 46
MindSpeed 5.1 47
Data Interchange 47
Reference Support 48
Object Support 48
GLOs 49
Virtual Machine 49
IEEE floating point 49
OE Profiler 50
User Settings 50

1

Introduction to VisualWorks 7

These release notes outline the changes made in the version 7 release of VisualWorks. Both Commercial and Non-Commercial releases are covered. These notes are not intended to be a comprehensive explanation of new features and functionality nor are they intended to be used in lieu of the product documentation. Refer to the VisualWorks documentation set for more information.

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.

Product Support Status

For the support status for versions of VisualWorks and Object Studio, refer to this web page:

> http://www.cincomsmalltalk.com:8080/CincomSmalltalkWiki/Cincom+Smalltalk+Platform+Support+Guide

ARs Resolved in this Release

The Action Requests (ARs) resolved in this release are listed in: ARs Resolved in VisualWorks 7.

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

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

Items Of Special Note

Store Database Format Changed

Changes to the database structure require Store users to update their existing database with:

```
Store.DbRegistry update7
```

After updating the database, it remains backward compatible with previous versions of Store.

VM Compatibility

Due to major changes in the virtual machine (object engine), such as are entailed by immutability and the microsecond clock, the VisualWorks 7 vms will not run earlier images. However, 5i.4b engines, incorporating many bug fixes that are present in the VW7 engines, will be provided in the near future. Watch the VisualWorks support site for updates.

New Toolset

A lot of conspicuous changes have been made in the toolset:

- The old browsers has been replaced with new browsers, based on the Refactoring Browser, by Refactory, Inc.
- A new parcel manager is introduced, improving the tool for finding and loading parcels.
- A new file browser.

Menus have been rearranged, making locations more reasonable.

The Application Developer’s Guide has been updated for the tools in the base.

Professional Debug Package

The Professional Debug Package from Crafted Smalltalk is included as an add-in in VisualWorks.

To load PDP, use the Parcel Manager, and select PDPLoadAll on the Essentials page. For further instructions, refer to its documentation in **`doc/ProfessionalDebug.pdf`**.

Note that if you can't find the **probes** menu (if it doesn't show up after loading the parcels, or if it disappears after loading Store), try switching the Look Policy, or spawn a new VisualLauncher.

Win95 support discontinued

Microsoft has discontinued support for the Windows95 operating system. Accordingly, beginning with VW 7 Cincom no longer supports Windows 95 as a VisualWorks platform. Support for previous releases on Win95 is subject to your support contract.

Pollock

A preview of the new GUI framework is included with this release. Refer to “New GUI Framework (Pollock)” later in these release notes for a description.

MacOS version requirements

MacOS versions 9.0.x through 9.2.x are now supported. VisualWorks may still run earlier versions, but are not supported, and may require assistance from Technical Support. In particular, a recent version of StdCLib is required. OS X support is provided as a preview.

File name lengths on Mac

The Mac HFS file system used on the CD for pre-OSX systems has a 31-character file name length limit, and so the files install with truncated names. You need to repair this after install. There are two options:

- If your system has HFS Plus, you can rename the files to their longer name.
- If your system has HFS, you need to modify any code referring to these files.

The files affected are:

```
com\
    ComEvent-EnabledApplicationExtensions.pcl
    ComEvent-EnabledApplicationExtensions.pst
    ComExternalDataDevelopmentEnhancements.pcl
    ComExternalDataDevelopmentEnhancements.pst
    ComSystemConfigurationServices.pcl
    ComSystemConfigurationServices.pst
```

```
goodies\
    other\
        DOME\
            parcels\
                WindowSystemCompatibility-changes.st
            tools\
                metadome\
                    help\
                        User-DefinedPropertySpecification.htm
        Objectivity\
            ObjectivitySmalltalkforVisualWorks.exe
    parc\
        WindowsDDE\
            Setting-upDDEFiletypeassociations.st
preview\
    opentalk\
        Opentalk-Load-PushBackClient.pcl
        Opentalk-Load-PushBackClient.pst
        Opentalk-Load-PushBackServer.pcl
        Opentalk-Load-PushBackServer.pst
        snmp\
            Opentalk-SNMP-Instrumentation.pcl
            Opentalk-SNMP-Instrumentation.pst
            snmpMIBs\
                standardMIBs\
                    IANA-ADDRESS-FAMILY-NUMBERS-MIB.txt
                    IANA-ADDRESS-FAMILY-NUMBERS-MIB.xml
```

HP-UX 10.2 VM availability

While we support HP-UX 10.2, the virtual machines for this platform are not on the CD. You can download the production HP-UX 10.2 VM from:

http://www.cincomsmalltalk.com/cstSummer2002/VM-HPUX10.tar.gz

2

VW 7 New and Enhanced Features

This section describes some of the biggest items to look for in this release.

Base system

New Collection methods

Several new methods have been added to Collection and its subclasses.

Also, isNotEmpty has been deprecated in favor of the ANSI notEmpty.

Dictionary

at: *aKey* **ifPresent:** *aBlock*

This method in Dictionary and KeyedCollection performs the opposite of at:ifAbsent:. The single argument *aBlock* is performed if *aKey* is found in the dictionary. The value passed to *aBlock* is the value of *aKey*. The method returns the value of *aBlock* if *aKey* is found, nil otherwise.

SequenceableCollection

runsFailing: *testBlock*

Evaluate *testBlock* with the receiver elements, selecting from the receiver “runs,” sequences of adjacent elements, for which the block returns false. Returns an OrderedCollection of those runs.

runsFailing: *testBlock* **do:** *enumerationBlock*

Same as runsFailing:, but evaluates *enumerationBlock* with each of those subsequences.

runsSatisfying: *testBlock*

Evaluate *testBlock* with the receiver elements, selecting from the receiver “runs,” sequences of adjacent elements, for which the block returns true. Returns an OrderedCollection of those subsequences.

runsSatisfying: *testBlock* **do:** *enumerationBlock*

Same as runsSatisfying:, but evaluates *enumerationBlock* with each of those subsequences.

piecesCutWhere: *testBlock*

Evaluate *testBlock* for successive pairs of the receiver elements, breaking the receiver into pieces between elements where the block evaluated to true. Returns an OrderedCollection of those pieces.

piecesCutWhere: *testBlock* **do:** *enumerationBlock*

Same as piecesCutWhere:, but evaluates *enumerationBlock* with each of those pieces."

Collection

fold: *aBlock*

Evaluate *aBlock* with two elements of the receiver, then with the result of the first evaluation and another element, then with the result of the second evaluation and another element, and so on. For unordered collections, elements are picked in some unspecified order, for ordered collections, they are picked in the order they are stored in the collection. Answers the result of the final evaluation. If the receiver is empty, fail. If the receiver contains a single element, answer the element.

groupedBy: *aBlock*

Return a Dictionary whose keys are the result of evaluating *aBlock* for all elements in the collection, and the value for each key is the collection of elements that evaluated to that key.

New Object methods

ifNil: *aBlock*

Returns the receiver if it is not nil; otherwise evaluates *aBlock* and returns the result.

ifNotNil:

Returns the receiver if it is nil; otherwise evaluates *aBlock* and returns the result.

ifNil: *nilBlock* **ifNotNil:** *notNilBlock*

If the receiver is not nil, answer the evaluation of *notNilBlock* (with the receiver as its argument if it takes one), otherwise answer nil. UndefinedObject redefines this to answer the value of *nilBlock*.

Instance Immutability

The system now provides the facility to make an individual object immutable. Attempts to assign to, become, or change the class of immutable objects will raise a NoModificationErrors exception. All literals and general instances of Number are now immutable.

This has extremely pervasive repercussions, so much so that a backwards compatibility parcel is provided that can force mutability for old code that relies on it. However, changing code to use the new facilities is quite straight-forward. Sending copy to an immutable object other than a general instance of Number answers a mutable copy. Hence code like:

```
'' writeStream
```

will no longer work until rewritten as:

```
'' copy writeStream
```

or:

```
(String new: N "some suitable initial size") writeStream
```

Apart from providing additional language safety through making literals immutable these facilities are useful in a number of contexts such as debugging (catching where an object is assigned-to), and for persistence, where attempts to modify are caught, retried and the updated objects written to persistent storage).

The new release of GemStone for VW7 uses these facilities, and is included in a non-commercial form as a goodie. Users of this feature for persistence and distribution schemes should study the GemStone code to see how to hook into the NoModificationError.

This is a very new area, and we expect the framework to evolve rapidly. Please use the vwnc list to communicate design ideas and issues with VisualWorks engineering.

New Weakness Facilities

The system now provides the ability to define arbitrary weak container classes, in addition to WeakArray. Any indexable pointer class can be defined such that its instances indexable fields are weak by using #weak as the value of the indexedType: keyword in the class definition.

Currently the system includes no additional examples, and we still need to implement a better garbage collecting scheme and provide backward-compatibility code for old WeakArray clients. The partial current implementation, however, allows you to begin to explore the possibilities.

Ephemerons

Along with the new weakness facilities, the system now supports "Ephemerons," which are special forms of Association that are used to attach properties to objects without preventing those objects from being garbage collected. They are like an Association whose key is weak, but they are more sophisticated in that references back to the key from the transitive closure of an ephemeron's other fields do not contribute to the key being counted as “alive” for gc purposes.

Both the DependentsFields dictionary, which associates dependents with instances that don't inherit from Model, and the EventHandlers dictionary that associates event handlers with all objects, have been reimplemented to use ephemerons. The result is that adding dependents or event handlers to an object will no longer prevent that object from being garbage collected, and means that code need no longer clean-up as carefully.

Ephemerons also support instance-based finalization, in that simply attaching an ephemeron to an object is a way of arranging that the ephemeron will be notified when there exist no other references to an object than from ephemerons. Due to time constraints we have yet to replace the postmortem finalization scheme based on WeakArray with a “last-rites” scheme based on ephemerons, but expect to in the next release.

Users of VSE may be familiar with ephemerons which first appeared there-in. We are indebted to George Bosworth for the design of ephemerons and to Barry Hayes for much of the garbage collector implementation.

TimeZone and Time

With the arrival of the object engine's microsecond clock, which provides the time in UTC (GMT) for all platforms, all users must now set the TimeZone. In particular, Mac and Windows NT users must now install a TimeZone, whereas they did not on previous releases. We intend to provide more automatic TimeZone selection facilities in the next release.

A new TimeZone instance creation method provides the ability to set different start and stop times for daylight savings time (DST). This is necessary, for example, in Europe, where start and stop times are referenced by UTC, and so are an hour different.

The Time class now derives its secondClock and millisecondClockValue from the microsecond clock, meaning that the millisecond clock no longer wraps every 49 days, but will not wrap for at least 18,000 years. This can affect code that assumes that the millisecond clock is a 32-bit integer, as did the old Random code. Users should check uses of millisecondClockValue to make sure code will not be broken by the wider clock.

External Interface exception handling

The system now allows one to catch machine-level errors in external interface calls such as bus errors or illegal instructions on Unix platforms as well as Windows. The error code returned from an external call that raised such an error is an Array of an exception code and the program counter at which the exception occurred. On Windows the exception code is the value of **`_exception_code()`** and on Unix is the signal number, e.g. SIGBUS.

One can also decide that exceptions should not be caught which can aid in debugging external code by allowing a platform debugger to trap the error rather than the VM. To turn off the catching of external errors do:

```
ObjectMemory registerObject: false withEngineFor: 'catchExternalCallErrors'
```

GUI Development

Many relatively small enhancements have been made to the GUI framework and tools:

- The UI painting tools continue to be improved.
- Hotkey support has been added for labels and group boxes, definable from the painting tools.
- The trigger Event system is now fully implemented for the widget subsystem, improving performance for cascading UI updates (documented in the *Application Developer's Guide*).
- Specifying fonts with the Named Font Selector has been improved for Windows and Unix fonts.
- Automatic support for the Wheel mouse in all scrolling views
- TableInterface methods columnFormats and columnFormats: have been replaced by methods elementFormats and elementFormats:.

UIPainter is TriggerEvent Driven

The trigger event system from VisualSmalltalk has been available as part of VisualWorks since version 3.0. With this release of VisualWorks, the trigger event framework has been given first class citizenship in the GUI system, and the framework has been expanded and enhanced to provide an easier path to use. The UIPainter, for example, is now fully event-driven.

Refer to the *Application Developer's Guide* for a description of the event system and how to use it. Also, see "Retiring the Polling UI" by Sam S. Shuster, which is available in the doc/TechNotes directory, and on the Documentation page of the Cincom Smalltalk Developer's Wiki:

http://www.cincomsmalltalk.com:8080/CincomSmalltalkWiki

ValueModel support for TriggerEvents

A modification to `Object>>changed:with:` enables value models to participate with the `TriggerEvent when:send:...` mechanism. If the changed aspect is a symbol, it is triggered as an event.

Dataset

The Dataset widget has been overhauled with improvements adapted with permission from the Aragon toolset. This includes optional row labels as buttons, optional row numbering, optional row and column lines, drag/drop column reordering, visual indication for column sorting, cursor flyby change when over column resizing area, optional column by column sort enablement. Additionally, when the column rows are selected to be viewed as buttons, this now shows directly in the UIPainter.

TreeView

The TreeView widget has been improved with these features:

- Optional full line selection (like the Hierarchical List).
- Optional Opened, Closed and Leaf item text emphasis selectors (exceeding capability of Hierarchical List)
- Item height now matches font

Window placement

A property of a canvas may now be set to determine the initial placement and size of a window.

- Window opening options expanded - on a per window basis one can choose from the following placement types:
  - Cascade, Window Center, Mouse Center, Specified Position.
  - Also, save size and save position support, automatic or via API.

Graphics

Base system support for reading XBM, JPEG, and GIF image formats. Loadable support is available for PNG format.

Graphic images may now be rotated an arbitrary number of degrees using the `Image` `rotatedByDegrees:` or `rotatedFastByDegrees:` messages. Prior to this an `Image` could be rotated only by increments of 90 degrees.

Tools

Parcel Manager

The Parcel Manager is a new tool for exploring the available parcels (loadable components of VisualWorks) and loading them. To start it, use the **System → Parcel Manager** item of the Visual Launcher menu, or press **F3** while in the Launcher window.

The Parcel Manager provides three ways of looking at parcels, corresponding to the three tabs on the left hand side of the window.

- **Suggestions** is a list of parcels organized by category, such as “Developer Tools” or “Graphics.” A category may include both supported VisualWorks components, and third-party contributions (goodies). (To distinguish supported parcels from goodies, they are shown using different icons).
- **Directories** shows the directories from the parcel search, organized hierarchically under three top-level items: “VisualWorks” for supported VisualWorks components, “Goodies” for third party contributions, and “Preview” for VisualWorks component still being worked on.
- **Loaded** shows parcels that are already loaded, again categorized.

When a parcel category or directory is selected, its parcels are displayed in the list on the right. The list can show the parcels arranged in either alphabetical order, or as a tree revealing the prerequisite relationship between them.

All lists support multiple selection, allowing the user to select multiple parcels from many categories or directories, to see their comments all at once.

File browser

VisualWorks 7 includes a new File Browser for working with files. The File Browser supersedes the File List of earlier versions of VisualWorks. To start it, select the **File → File Browser** in the Visual Launcher menus, click its button on the toolbar, or press **F2** while the Launcher window is active.

By default, the File Browser selects the current working directory when it opens. This usually takes only a few seconds. However, some users report longer opening times (the time depends on the platform and on the directory tree structure). If a long opening time is a problem, you can use the **Directory → Initial Selection** menu to change this preference.

The old File List tool is no longer available in the Launcher, but is still present in the image in this release of VisualWorks. If you wish to use it, you can open it by evaluating the following expression:

```
Tools.FileBrowser open
```

Browsers

The Refactoring Browser, by Refactory, Inc., is now fully integrated with the VisualWorks system, and is the basis of the new system browser. In addition to providing powerful refactoring and code checking facilities, the new browser provides a more uniform interface to standard browser operations.

The Application Developer’s Guide and some other documents have been updated to reflect the new browser.

There are new icons that may need some explanation, and is provided as a pick on the system browser help menu.

The old browsers are deprecated, but are still present in the system, though not available from the VisualLauncher. Refer to the “Tools” section in the Deprecated chapter for more information.

Workspace

The Workspace now allows you to select the encoding of a file, when used to edit files. The default encoding is Source (UTF-8). To change the encoding for a particular workspace, select **Edit → Encoding**.

WebService enhancements

Included in the NetClients is support for XML-base web transactions using SOAP, WSDL and UDDI.

Client and server support for SOAP is currently separated along possibly unintuitive lines.

Core frameworks for SOAP/WSDL client support are provided as part of Net Clients, and parcels are in the **`net/`** directory. These are documented in the *Internet Client Developer's Guide*.

For server-side support, these core frameworks are integrated with the Opentalk general server architecture. For SOAP server implementation, you must load the Opentalk-SOAP parcel. Refer to the *Opentalk Communication Layer Developer's Guide*, the “Opentalk SOAP” chapter, for additional information.

Secure Socket Layer (SSL)

SSL functionality is now released and supported. The SSL parcel provides implementation of Netscape's SSL protocol version 3.0. The implementation of the protocol itself is complete, limited only by the number of available encryption algorithms and the implementation of X509 certificates, which is being developed concurrently as well.

Refer to the *Internet Client Developer's Guide* for detailed information.

Database

Threaded ODBC EXDI

ODBCThapiEXDI has been promoted to fully supported product, out of beta.

Oracle Unix/Linux Interface

To use OracleLinuxInterface (and the thapi equivalent), it is no longer necessary to define $LD_LIBRARY_PATH on Linux systems in order to use the Oracle interfaces. The OS environment variable, ORACLE_HOME, is used to locate the shared library.

Oracle 8/9 support

The OracleEXDI in prior releases uses Oracle OCI 7. In VW 7, OracleEXDI uses Oracle OCI 8 interface. To make upgrading from the old to the new OracleEXDI easier, and at the same time, to allow the co-existence of Oracle 8 and Oracle 7 EXDI, we renamed old OracleEXDI and its classes from Oracle* to Oracle7* (for instance, `OracleConnection` to `Oracle7Connection`). The new OracleEXDI and its classes use Oracle* names (e.g., `OracleConnection`).

If you are using Oracle client later than 8, you can use the new OracleEXDI in the same way as before and take the advantage of the performance from Oracle OCI 8. However, if you are still using an Oracle 7.x client, you have to use Oracle7* classes and make changes to your existing applications accordingly.

If your Oracle library is not being found, try these suggestions, which might resolve the problem.

- If you have Oracle 7.x client, the prior OracleEXDI library has been renamed as Oracle7EXDI and moved to **`obsolete/database/`**. Use this library to connect to Oracle 7 databases, because the new library uses the Oracle 8 OCI, so it cannot be used to connect to 7 databases.
- If you have Oracle 8, using Windows (98/NT/2K), you may need to change the name of the DLL in `OracleNTInterface` (and Win95). VW 7 now officially conforms to the Oracle 8.x api, and the DLL names have been changed accordingly.
    - If you have Oracle 8.0 through 8.03, then you may need to replace '`oci.dll`' in the interface with whatever DLL is present, such as, '`ora803.dll`'. That's the general pattern.

- If you have Oracle 8.04 or higher, look for a DLL named **`oraclient8.dll`** (e.g., **`c:\orant\bin\oraclient8.dll`**). If your OracleNTInterface has ‘oci.dll', try replacing that with 'oraclient8.dll'. Some users report better results using this dll, in spite of Oracle's recommendation to use 'oci.dll'.

- If you are using Solaris and have Oracle 8.0.4 client, you probably need to insert an extra library file, 'scorept.so' ahead of 'libclntsh.so' in the OracleSolarisInterface pool variables area. There was a missing symbol in the usual shared library, **`libclntsh.so`**, and **`scorept.so`** has the missing symbol. If you cannot find this file in your **`oracle/8.0.4/lib`** directory, your DBA should be able to acquire it. (“Library not found” errors on Windows platforms.)

XML enhancements

XML Schema is now out of beta and is a supported feature. Refer to the *Application Developer’s Guide* for documentation.

The foundational XML support has received several enhancements:

- improved XML parser compliance
- SAX extended for writing XML files
- support for XPath (load the XPath.pcl parcel)
- XSL upgraded to substantially satisfy the 1.0 spec

VisualWave/Web ToolKit

Startup

In 5i.4, when loading into a generic headless image, there could be initialization errors. In particular, if a parcel MyWebApp.pcl had the Web Toolkit parcel as a prerequisite and was loaded via the command-line into a generic run-time image, the Web Toolkit startup would run as soon as its parcel was fully loaded. This includes reading configuration files, so an action in the configuration files which referenced classes in MyWebApp would fail, because those classes were not yet present.

In version 7 this has been partially addressed by performing lazy initialization of the Web Toolkit when the first HTTP request is received. Note that this can make the first HTTP request to a new image much slower than subsequent requests, and that testing of configuration files requires at least one request to be sent to the server.

Note also (this has not changed since 5i.4) that the startup sequence differs between development and runtime images.

At the last minute we realized the fix is not complete. The following work around removes the call to #configureServer in WebConfigurationManager class method returnFromSnapshot. Edit this method as follows:

```
returnFromSnapshot
    WebSite clearAllCaches.
    "If we're in a Development environment, just restart logging."
    (ProcessEnvironment isDeveloping)
        ifTrue: [ WebConfigurationManager initializeLoggingChannel ].
```

In a VisualWave or Web Toolkit image packaged with runtime packager, many of the errors reported in the Transcript log (hlst.tr) are not really errors, but the result of invalid parsing attempts by the WebSettings class. The standard command line entries are parsed and handled correctly elsewhere.

Configuration

The configuration process has been significantly enhanced. The configuration pages have been reorganized, and the demo applications are now easier to get to. This includes automatically loading any required parcels and links directly to the demos from the configuration pages.

The response to configuration errors is now much more robust. Configuration errors do not disable the entire server, and it is still possible to access the configuration site to diagnose and correct errors even with significant errors in the site. Configuration errors are now highlighted and the corresponding web sites are de-activated.

Security

A security loophole in 5i.4, where an SSP URL that ended with a period would return the source code of the SSP, has been closed.

Logging

A number of problems with logging have been resolved, relating to contention when switching logs, renaming the log file, or saving the image. In addition, more system events are logged, most notably configuration reset events, so it's possible to tell the set of configuration files in use by looking at the log.

Debugging

Response to some errors has been improved. In particular, trace backs when a file is not found now list the name of the file, making it much simpler to determine the problem.

The Headless facility now includes a preliminary version of a becomeHeadfull method. This should be used with caution, since it may not correctly switch over everything in the image, particularly if there are user actions that need to be run. However, if used for debugging purposes it can be very useful, switching a headless runtime image into a mode where it can be more easily debugged.

It is not possible to save a runtime image in debug mode. By default a Web Toolkit development image starts in debug mode, meaning that, for example, exceptions result in debuggers, that SSP compiled representations are not cached. By default a packaged image, built using the Web menu's **Save Production Runtime As...** option is in production mode, so exceptions are caught and logged and compiled representations are cached. Using the Web menu's **Save Headless Development Image As...** you can save a headless image which is still in debug mode. This is equivalent to the regular **Save Headless As...** normally available when the Headless parcel is loaded. This may make it easier to debug problems that occur only in a headless runtime.

A useful goodie to be aware of is the MultiTimeProfiler, contributed by Florin Mateoc and available in the ATProfilingEnhancements parcel. This allows you to profile programs whose work is distributed amongst many processes.

Demos

The demos have been enhanced, most notably the ToyzInc. demo. This now includes better laid-out web pages, and also demonstrates some of the more advanced facilities in the system, including serving static components (e.g. images) differently from the dynamic pages. For a discussion of some of the design philosophy used in this example, see the article "Objects and the Web", from *IEEE Software*, March/April 2002, also available at:

http://www.cincom.com/newsmalltalk/community/pdf/s2knight.lo1.pdf

Documentation

Documentation has been significantly upgraded, reorganized, and renamed (see “Documentation” below).

Terminology has been slightly changed to clarify the different sorts of web development tools within VisualWorks. The Application Server as a whole is now referred to as the VisualWorks Application Server. Three distinct server technologies are supported: Smalltalk Server Pages, Servlets, and VisualWave. The name "VisualWave", which was formerly used to refer to the entire product, is now only used to refer to a single server technology.

Class categories have been updated for better consistency and to reflect the different product categories.

Known limitations

Tiny HTTP Configuration

The Tiny HTTP Server, like the IPWebListener, asks you to specify a hostname when creating the server. However, the Tiny HTTP Server has the limitation that this hostname is always used, regardless of the actual name used to access the server. For simple requests the name doesn't matter, but if the server needs to generate a URL (e.g. for a redirection) then the name becomes significant. In particular, the default name of "localhost" will only work for clients running on the same machine as the server. If you access the Tiny HTTP server from another machine, then these redirections will not work. We suggest using the proper machine name or IP address as an entry in this configuration (e.g. "aknightnt" or "192.168.0.1"). The IPWebListener does not have this limitation.

Operating System Limitations

Running a web server can be demanding on the underlying socket and threading support in an operating system, even for limited usage. Most notably, the socket and thread implementations in the Windows NT family of operating systems (including Windows 2000 and Windows XP) are significantly more scalable and reliable than those in the Windows 98 family (including Windows ME). While it is possible to develop web applications using Windows 98 or ME, these may experience intermittent problems, particularly when also running the client on the same machine as the server. Similar concerns apply on the Mac, where MacOS X has significant improvements over MacOS X. In our tests of the web functionality we find better results using the MacOS X preview VM than the MacOS 9 VM.

Loading Wave Palette Icons

If you load VisualWave Server or the Web Toolkit, then load UIPainter, the VisualWave-specific painter icons are not properly installed. To correct this, you need to load the WaveUIPainterTools parcel. (AR43320)

Case Sensitivity in Web Toolkit Pages

All names in Web Toolkit are currently case sensitive (e.g., form parameter names). This is true even in ASP-style pages (i.e., files that end in **`.asp`**). This is inconsistent with Microsoft's ASP which is not case-sensitive. (AR44179)

Buffering

Buffering of responses is currently not supported. This means that all headers must be set before *any* output is sent to the response. (AR44143)

Authorization with CGI Gateway

The CGI gateway does not correctly pass back the information for an authorization challenge (401). This can lead to authorization challenges being rejected without the opportunity to enter a password, or an entered password not being properly passed to the application server. (AR44336)

A workaround, using the CGI gateway, is to use an authentication challenge scheme other than the HTTP 401 response. For example, requests from users not yet logged in could be redirected to a login page. This would require either using a standard block of code at the beginning of each page, or modifying the CheckAuthorizationServlet to do this redirection instead of issuing a 401 response.

As another alternative, use a different gateway (i.e. ISAPI or FastCGI) or the TinyHTTP server.

Startup Directory and Configuration Files

Note that the web toolkit starts the search for configuration files in the working directory of the application. Under Windows, this working directory may be unexpectedly set to unusual values, including the root directory of the drive (C:\) the windows system directory, or others, depending on how the application was launched.

To avoid this, you can use an absolute path name, possibly one that includes an environment variable. Note that the default configuration files use the $VISUALWORKS environment variable in order to locate the files reliably.

Alternatively, you can take steps to ensure that the working directory is set to the expected value. If using a Windows shortcut, specify the "Start In:" directory. If starting from a command-line, ensure that your working directory is the one you expect for your configuration files.

For an NT or Windows 2000 service, your registry entry should contain a string parameter "AppDirectory" to reference the working directory so you do not need a fully qualified path to the image in the "Application" parameter. The two Parameters for your service should look like:

Name	Type	Data
AppDirectory	REG_SZ	C:\VWHeadless
Application	REG_SZ	C:\VWHeadless\vwnt.exe -noherald runtime.im

Store

We are dropping the funny typography, so will now refer to “Store” rather than “StORE.”

Expanded documentation

The Store documentation has been removed from the Application Developer’s Guide and turned into a separate document, the Source Code Management Guide (SourceCodeMgmtGuide.pdf). It is currently in a draft form, and will continue to be expanded during the next release cycle. Watch the Cincom Smalltalk Documentation page for updates.

New Connection dialog

The connection dialog now allows you to specify the table owner. This is necessary when multiple Store repositories exist in a single database. The table owner selects the specific repository.

Prerequisite version control

Additional properties on packages, bundles and parcels allow better control of versions that are acceptable to satisfy prerequisites. View the help on the properties pages, and the *Source Code Management Guide* for more information.

Binary load enhancements

A new option in Store Settings dialog, on the **Load Options** page, allows you to suppress binary loading of packages published binary. Selecting this option forces loading and compiling from the source code version.

A variety of problems with binary loading, especially for reloads, have been solved, making Store more reliable.

Prerequisite specification

New properties allow better control of prerequisites, such as testing for acceptable versions based on the version strings.

Performance enhancements

A number of changes have been made to the system to improve Store performance. In particular, loading from optimized databases is up to 40 times faster, and updating, reconciling, and comparisons are up to 30 times faster.

Bundle content management

A new menu option (**Republish bundle…**) allows changes to bundle blessing, comment, and properties for bundles that are not loaded. Access the option from the Versions browser (**Package → Versions → List** in the system browser, then **Package → Republish bundle…**), or from the Published Items list (**File → Republish bundle…**).

Within the republish dialog your are allowed to change the bundle content and order (load order), and to add and remove components. Use the Contents page of the dialog to make these changes.

Merge tool and bundles

The Merge Tool now shows differences between bundle contents and properties.

Database table alias support

Added support for database table aliases, for better 3rd party back-end support.

Known Limitations

Bundle and override usage

Many customers have encountered difficulty with using overrides, especially overrides in bundles. Some of this is attributable to bugs in the override system, many of which are corrected in this release.

Other problems were caused by misunderstanding of the intended behavior of overrides within a bundle. In particular, overridden definitions are not preserved between packages and bundles contained within a bundle; only the last loaded definition is preserved.

Please refer to the discussion in the introductory chapter of the new *Source Code Management Guide* for a full discussion.

StoreForSQLServer incompatibility with ODBC

If you have loaded StoreForSQLServer in your image, and you do a SELECT from a TEXT column type, using the AdHocSQL Tool or from your own application code, you may get a UHE when retrieving data. An instance variable (unicodeType) may not have been initialized.

The method ODBCLargeObjectBuffer >> descriptor:position:session: in the StoreForSQLServer parcel is not fully compatible with the ODBC parcel, leaving an instance variable uninitialized. The regular ODBC parcel does not implement this method in ODBCLargeObjectBuffer.

A patch will soon be released with the fix. The interim fix is to simply remove the descriptor:position:session: method from the StoreForSQLServer parcel, or from your image. This allows the super class method to take over, as with the regular ODBC parcel.

Opentalk

Exception forwarding

Formerly, STST forwarded just the error string from any server error and raised an instance of OtServerError in the client thread, which made error handling a lot different than in non-distributed cases, thus hurting transparency a lot.

In VW7 we have replaced this with transparent exception forwarding. This means that the exception raised on the server will be forwarded and reraised in the client image.

Note that there is a limitation that the forwarded exception cannot be resumed. Support for resumption of forwarded exceptions would incur heavy performance penalty which is not worth it in a production system (the necessary infrastructure would be as complex as the infrastructure used to support remote debugging).

Catching Broker Errors

By default the server process loop handles any errors by generating an error event, and resuming the loop expecting the next message. However unless the user application explicitily registers an interest in receiving these broker events, the server process events will just fall off the table, causing the errors to be silently suppressed. The result of this is that the developer often sees only side-effects of those suppressed errors, e.g., timed-out requests or requests returning with "connection closed," if the client or server broker isn't shutting down early enough.

To aid with catching these errors during development a standard set of event handlers was wrapped in a convenient API. The API methods are following:

handleErrors (default)
: Uninstall any of the event handlers added by the other methods

passErrors
: Install a handler passing the exception, i.e. a UHE notifier will pop up in the server thread

haltErrors
: Evaluate 'self halt' on any error event

showErrors
: Log error events in the Transcript

Opentalk process priorities

Priorities of the background processes created by Opentalk brokers have been lowered. As of this release, none of them are running in the critical process priority range (LowIOPriority and above). Priority configuration parameters were also added to allow broker performance tuning.

The following adaptor configuration parameters were added (applicable for connection-oriented adaptors):

listenerPriority

listenerBacklog

The following transport configuration parameters were added:

workerPriority

serverPriority

The following API in ConnectionListener was deprecated to encourage switching to configuration parameters instead:

listenerProcessPriority

listenerProcessPriority:

Resumable timeout exceptions

Request timeout exceptions are now signaled as a special subclass of OtECommunicationFailure, called OtETimeout. In contrast with the generic OtECommunicationFailure, the timeout exception is resumable, when resumed it makes the client thread wait for another timeout period. This scheme is aimed at providing better support for long lasting computations possibly exceeding the default request timeout settings.

DST

Demo parcel renamed

The DST example classes mentioned in the documentation are now in parcel DST_Sample, not in DST_Demo as the documentation presently claims. DST_Sample now contains all those example and sample applications that we are not moving to 'obsolete'.

Installation in the presence of Store

If you load Store before loading DST, you cannot load DST using the DST_Install parcel. Instead, load one of the following parcels, to load an appropriate set of DST parcels:

- DST_COS-Services - DST Services withou tools or GUI
- DST_I3 - DST with I3 support,
- DST_Tools_Development - DST with the usual DST tools
- DST_Sample - all of the above plus the examples mentioned in the DST docs.

Virtual Machine

Backward compatibility

Due to extensive changes in the VisualWorks 7 virtual machines, 5i.4 and earlier images cannot be run on VW 7 VMs. 5i.4b VMs, incorporating several bug fixes, will be available soon.

New debugging engines

New “assert” engines have been added to the set of VMs, These engines are fully-optimized, but with asserts compiled-in and enabled. They run at least 50% of the speed of the fully-optimized production engine, even though they check engine asserts. For normal development, this provides perfectly acceptable performance while checking the engine during normal use. They are named **`vwPlatformNameast`**, for example **`vwlinux86ast`** and **`vwntast.exe`**.

Extensions for Base Support

Several extensions to the Base depend on new facilities in the VM, such as:

- Microsecond clock
- Instance immutability
- Ephemerons
- Generalized weakness

For more information, refer to the "Base system" section.

Packaging as an Executable

The facilities for packaging a VM and image in a single executable for Windows and Mac platforms have been moved out of beta into supported release. Instructions for doing this are in text files in the **`packaging/win/`** and **`packaging/mac/`** directories.

Documentation

By request, significant documentation updates will be listed here.

Comprehensive document index

Piloted in 5i.4 is a comprehensive index of PDF documents. This employs a feature of Adobe Acrobat, and you must use a version of the Reader that supports the indexing feature. The latest version of the Reader supports this on most platforms supported by VisualWorks.

PDF renaming

We have renamed the PDF document files to make the file names more recognizable. You can still use Welcome.pdf as an index to documentation.

Browser documentation

The introduction of the new browser, based on the Refactoring Browser, has entailed extensive changes to the Application Developer's Guide, which has been updated and expanded to cover this tool and its use. Other documents have been updated as possible, but lag behind.

Store documentation

Store documentation has been removed from the Application Developer's Guide and made its own document, the *Source Code Management Guide*. The documentation is expanded, but still incomplete. Watch for updates on the Cincom Smalltalk Documentation site.

Opentalk

The *Opentalk Communication Layer Developer's Guide* has a new chapter describing Opentalk SOAP support.

XML framework

A chapter describing the XML support framework has been added to the *Application Developer's Guide*.

WebServices

WebServices, which includes SOAP, WSDL, and UDDI frameworks, has been added the *Internet Client Developer's Guide*. Also, a chapter on SSL support has been added.

NetClients

The *Internet Client Developer's Guide* has been expanded and renamed, as the Network Client Developer's Guide. Updates to several areas are included, notably in MIME. The email chapter has been revised and corrected, providing much more complete and accurate information about developing an email client application.

Events

Documentation of the event system (formerly known as sysdeps, or the VSE event system, or TriggerEvents) has been included. The *Application Developer's Guide* includes a section describing how to define and evoke events. The widget descriptions in the *GUI Developer's Guide* have been expanded to include the events they trigger.

VisualWave/Web Toolkit

The VisualWave documentation has been enhanced to complete its coverage of the new WebToolkit functionality, first introduced in 5i.4.

The main document has been split into two, separating coverage of different types of web application development:

Web GUI Developer's Guide is intended to guide VisualWorks programmers in creating web-based applications using VisualWave. VisualWave is a server technology supported by the VisualWorks Application Server.

Web Application Developer's Guide is designed to help VisualWorks programmers create Web applications using the VisualWorks Application Server.

Help

A short document describing how to write your own help documents is available on the web. See VW Help System in the **`doc/TechNotes`** directory.

An Examples Catalogue help topic has been added to the help system, providing a central location to find examples included with VisualWorks.

Tools help has been updated to describe the new browser. Also, contextual links into the help from the browser have been added.

3

Deprecated Features

By deprecating certain features, we remove them from the system. These are made available for a limited time as parcels in the **`obsolete/`** directory, to provide you the opportunity to port applications away from using the features before they are removed altogether. This directory is on the default parcel path.

Base System

isNotEmpty

The selector isNotEmpty has been deprecated in favor of the ANSI notEmpty.

Database

Oracle 7 EXDI

The old OracleEXDI parcels have been renamed Oracle7EXDI and moved to the **`obsolete/db`** directory. In the future, Oracle 7 support will be removed entirely.

DST

These example parcels have been replaced by DST_Sample, but are still available in **`obsolete/dst`**:

DST_Demo
DST_Demo_IDL
DST_Protocols_NCS
DST_Tools_Profile

Also, the pixmap and icon directories associated with DST_Demo have been moved to **`obsolete/dst`**. Those icons used by the DST Repository Browser have been retained in **`dst/`**.

GUI

TableInterface methods columnFormats and columnFormats: have been deprecated in favor of methods elementFormats and elementFormats:.

Tools

In the presence of the new system browser, based on the Refactoring Browser, the old browsers have been deprecated, but are still present in the system. These browsers are regarded as deprecated, and will be removed from a future release.

If you want or need to use one of the old browsers, you can open them by evaluating these expressions:

Package Browser	TabApplicationSystemBrowser openOnPackages.
System Browser	FullNotebookSystemBrowser openOnSystem
Parcel Browser	FullNotebookSystemBrowser openOnParcels
Class Browser	TabApplicationSystemBrowser openOnClass: aClass
Hierarchy Browser	TabApplicationSystemBrowser openOnClassHierarchy: aClass

4

Preview Components

Several features are included in a `preview/` and available on a “beta test” basis. This is a renaming of the directory from prior releases, and reflects looser criteria for inclusion, allowing us to provide pre-beta quality, early access to forthcoming features. Several are described in the following sections. Browse the directory contents for last minute inclusions.

New GUI Framework (Pollock)

Over the last several years, we have become increasingly dissatisfied with both the speed and structure of our GUI Frameworks. In that time, it has become obvious that the current GUI Frameworks have reached a plateau in terms of flexibility. Our list of GUI enhancements is long, supplemented as it has been by comments from the larger VisualWorks communities on comp.lang.smalltalk and the VWNC list. There is nothing we would like more than to be able to provide every enhancement on that list, and more.

But, the current GUI Frameworks aren't up to the job of providing the enhancements we all want and need, and still remain maintainable. In fact, we are actually beyond the point of our current GUI frameworks being reasonably maintainable.

This is not in any way meant to denigrate the outstanding work of those who created and maintained the current GUI system in the past. Quite the opposite, we admire the fact that the existing frameworks, now over a decade old, have been able to show the flexibility and capability that have allowed us to reach as far as we have.

However, the time has come to move on. As time has passed, and new capabilities have been added to VisualWorks, the decisions of the past no longer hold up as well as they once did.

Over the past several decades, our GUI Project Leader, Samuel S. Shuster, has studied the work of other GUI Framework tools including, VisualWorks, VisualAge Smalltalk, Smalltalk/X, Dolphin, VisualSmalltalk, Smalltalk MT, PARTS, WindowBuilder, Delphi, OS/2, CUI, Windows, MFC, X11, MacOS. He has also been lucky enough to have been privy to the “private” code bases and been able to have discussions with developers of such projects as WindowBuilder, Jigsaw, Van Gogh and PARTS.

Even with that background, we have realized that we have nothing new to say on the subject of GUI Frameworks. We have no new ideas. What we do have is the tremendous body of information that comes from the successes and failures of those who came before us.

With that background, we intend to build a new GUI Framework.

High Level Goals

The goals of the new Framework are really quite simple. Make a GUI Framework that maintains all of the goals of the current VisualWorks GUI and is flexible and capable enough to see us forward for at least the next decade.

We add must add additional, less concrete goals too:

- The new GUI Framework must be more accessible to both novice and expert developers.
- The new GUI Framework must be more modular.
- The new GUI Framework must be more adaptable to new looks and feels.
- The new GUI Framework must have comprehensive unit tests.

Finally, and most importantly:

- The new GUI Framework must be developed out in the open.

Pollock

The name for this new Framework has been “code named” Pollock after the painter Jackson Pollock. It's not a secret code name. We came up with the name during our review of other VisualWorks GUI frameworks, most directly, Van Gogh. It's just our way of saying we need a new, modern abstraction.

Pollock Requirements

The high level goals lead to a number of design decisions and requirements. These include:

No Wrappers

The whole structure of the current GUI is complicated by the wrappers. We have Spec wrappers, and Border wrappers, and Widget wrappers, and Bounded wrappers and more. There is no doubt that they all work, but learning and understanding how they work has always been difficult. Over the years, the wrappers have had to take on more and more ugly code in order to support needed enhancements such as mouse wheel support. Pollock will instead build the knowledge of how to deal with all of these right into the widgets.

No UIBuilder at runtime

The UIBuilder has taken on a huge rule. Not only does it build your user interface from the specification you give it, it then hangs around and acts as a widget inventory. Pollock will break these behaviors in two, with two separate mechanisms; a UI Builder for building and a Widget Inventory for runtime access to widgets and other important information in your user interface.

New Drag/Drop Framework

The current Drag/Drop is limited and hard to work with. It also doesn't respect platform mouse feel aspects, nor does it cleanly support multiple window drag drop. Pollock will redo the Drag/Drop framework as a state machine. It will also use the trigger event system instead of the change / update system of the current framework. Finally, it will be more configurable to follow platform feels, as well as developer extensions.

The Default/Smalltalk look is dead

We will have at the minimum the following looks and feels: Win95/NT, Win98/2K, MacOSX and Motif. We will provide a Win2K look soon after the first production version of Pollock.

Better hotkey mapping

Roel Wuyts has been kind enough to give permission allowing us to use his MagicKeys hot key mapping tool and adapt it for inclusion in the base product. Thank you Roel.

XML Specs

We will be providing both traditional, array-based, and XML based spec support, but our main format for the specifications will be XML. We will provide a DTD and tools to translate old array specifications to and from the new XML format. Additionally, in Pollock, the specs will be able to be saved to disk, as well as loaded from disk at runtime.

Conversion Tools

With the release of the first production version of the Pollock UI Framework, we will also produce tools that will allow you to convert existing applications to the new framework. These tools will be in the form of refactorings that can be used in conjunction with the Refactoring tools that are now a integral part of VisualWorks, as well as other tools and documentation to ease the developer in transitioning to the new framework.

Unit Tests

Pollock will and already does, have a large suite of unit tests. These will help maintain the quality of the Pollock framework as it evolves.

New Metaphor

The Pollock framework is based on a guiding metaphor; “Panes with Frames, with Agents and Artists.” More on that below.

Automatic look and feel adaptation

In the current UI framework, when you change the look and/or feel, not all of your windows will update themselves to the new look or feel. In Pollock, all widgets will know how to automatically adapt themselves to new looks and feels without special code having to be supplied by the developer. This comes “free” with the new “Panes with Frames, with Agents and Artists” metaphor.

The New Metaphor: Panes with frames, agents, and artists

In Pollock, a Pane at its simplest is akin to the existing VisualComponent. A Pane may have subpanes. There will be an AbstractPane class. The Window is also a kind of Pane, but because we don't plan to re-write the whole world, it will remain in it's own hierarchy. Also, the Screen becomes in effect the outermost Pane. Other than that, all panes (widgets) will be subclassed in one way or another from the AbstractPane.

The Frame has a couple of pieces, but in general can be thought of as that which surrounds a pane. One part of a Frame is its layout. That is like our existing layout classes, that which defines where it sits in the pane that encloses it. It optionally may have information about where it resides in relation to sibling panes (and their Frames).

A border or scroll bar in the pane may “clip” the view inside the Pane. In this case, the Frame also works as the view port into the pane. As such, a pane may be actually larger than its Frame, and the Frame then could provide the scrolling offsets into the view of the Pane. The old bounds and preferred bounds terminology is gone, and replaced by two new, more consistent terms: visible bounds and displayable bounds. The visible bounds represents is the whole outer bounds of the pane. The displayable bounds represents that area inside the pane that is allowed to be displayed on by any subpane. For example, a button typically has a border. The visible bounds is the whole outer bounds of the pane, while the displayable bounds will represent that area that is not “clipped” by the border.

Another example is a text editor pane. The pane itself has a border, and typically has scroll bars. The visible bounds are the outer bounds of the pane, and the displayable bounds are the inner area of the text editor pane that the text inside it can be displayed in. The text that is displayed in a text editor, may have its own calculated visible bounds that is larger than the displayable bounds of the text editor pane. In this case, the Frame of the text editor pane will interact with the scroll bars and the position of the text inside the pane to show a view of the text.

Artists are objects that do the drawing Pane contents. Note: No longer does the “view” handle all of the drawing. All of the `displayOn:` messages simply get re-routed to the Artist for the Pane. This allows plugging different Artists into the same Pane. For instance, a Text Pane could have a separate Artist for drawing word-wrapped and non-word-wrapped text. A “Composed Text Pane” could have a separate artist for viewing the text composed, as well as maybe in XML format. Additionally, the plug and play ability of the Artist allows for the automatic updating of panes when the underlying look changes. No longer will there be multiple versions of views or controllers, one for each look or feel. Instead, the Artists and Agents will, when needed, be able to be plugged directly into the pane.

Agents are that which interact with the Artist and the Pane on behalf of the user. Now, if this sounds like a replacement of the Controller, you're partially correct. In the Pollock framework, the Controllers will have much less “view” related behavior. Instead, they will simply be the distributor of events to the Agent via the Pane. This means that our Controllers, while they'll still be there, will be much more stupid, and thus, able to be much less complex and less coupled to the Pane. Like the Artist, the Agent is pluggable. Thus, a TextPane may have a read-only Agent, which doesn't allow modifying the model.

Other notes of interest

The Change/Update mechanism will be taking a back seat to the TriggerEvent mechanism. The ValueModel will still remain, and Pollock will be adding a set of TriggerEvent based subclasses that will have changed, value: and value events. Internal to the Pollock GUI, there simply will not be a single place where components will communicate with each other via the change/update mechanism as they do today. While they will continue to talk to the Model in the usual way, there will be much less chatty change/update noise going on.

The ApplicationModel in name is gone. It was never really a model, nor did it typically represent an application. Instead, a new class named UserInteface replaces it. This new class will know how to do all things Pollock. Conversion tools will take existing ApplicationModel subclasses and make UserInterface subclasses.

A new ScheduledWindow class (in the Pollock namespace) with two subclasses: ApplicationWindow and DialogWindow. The ScheduledWindow will be a full-fledged handler of all events, not just mouse events like the current ScheduledWindow. The ApplicationWindow will be allowed to have menus and toolbars, the ScheduledWindow and DialogWindow will not. The ApplicationWindow and DialogWindow will know how to build and open UserInterface specifications, the ScheduledWindow will not. Conversely the UserInterface will only create instances of ApplicationWindow and DialogWindow.

So, What Now?

The work on Pollock has already started. In the VisualWorks 7 distribution, we have provided a very basic beta framework. The goal of the first beta is very simple: A Window that has a Label and an Icon. A Button that has a Label and an Icon.

The next milestone is VisualWorks 7.1. For that, we hope to have all of the basic widgets done: InputField, TextEdit, CheckBox, RadioButton, Label, ComboBox, List, GroupBox and Divider. Additionally, a UIPainter tool and all of the basic supporting builders to create windows with these widgets. Also, many of the secondary widgets (TreeView, Dataset/Table, SpinButton, Resizer, SubPane, TabControl, Slider, ProgressBar, etc) will be in progress.

Following that is VisualWorks 7.2. For that, all of the widgets will be done and complete. All of the tools completed. Additionally, tools and utilities for converting existing GUIs to run on Pollock. Pollock will co-reside in the image along side the existing GUI framework. Finally, the Pollock GUI Painter will be available.

After that, it's on to migrating our own tools and browsers to Pollock. Followed in time by the obsoleting of the old GUI framework to a compatibility parcel.

Opentalk SNMP

SNMP is a widely deployed protocol that is commonly used to monitor, configure, and manage network devices such as routers and hosts. SNMP uses ASN.1 BER as its wire encoding and it is specified in several IETF RFCs.

The Opentalk SNMP preview partially implements two of the three versions of the SNMP protocol: SNMPv1 and SNMPv2. It does so in the context of a framework that both derives from the Opentalk Communication Layer and maintains large-scale fidelity to the recommended SNMPv3 implementation architecture specified in IETF RFC 2571.

Usage

Initial Configuration

Opentalk SNMP cares about the location of one DTD file and several MIB XML files. So, before you start to experiment, be sure to modify 'SNMPContext>>mibDirectories' if you have relocated the Opentalk SNMP directories.

Broker or Engine Creation and Configuration

In SNMPv3 parlance a broker is called an “engine”. An engine has more components that a typical Opentalk broker. In addition to a single transport mapping, a single marshaler, and so on, it must have or be able to have

- several transport mappings,
- a PDU dispatcher,
- several possible security systems,
- several possible access control subsystems,
- a logically distinct marshaler for each SNMP dialect, plus
- an attached MIB module for recording data about its own performance.

So, under the hood, SNMP engine configuration is more complex than the usual Opentalk broker configuration. You can create a simple SNMP engine with

```
SNMPEngine newUDPAtPort: 161.
```

But, this is implemented in terms of the more complex method below. Note that, for the moment, within the code SNMP protocol versions are distinguished by the integer used to identify them on the wire.

```
newUdpAtPorts: aSet
| oacs |

oacs := aSet collect: [ :pn |
    AdaptorConfiguration snmpUDP
        accessPointPort: pn;
        transport: ( TransportConfiguration snmpUDP
            marshaler: ( SNMPMarshalerConfiguration snmp ) )].

^(( SNMPEngineConfiguration snmp )
    accessControl: ( SNMPAccessControlSystemConfiguration snmp
        accessControlModels: ( Set
            with: SNMPAccessControlModelConfiguration snmpv0
            with: SNMPAccessControlModelConfiguration snmpv1 ) );
        instrumentation: ( SNMPInstrumentationConfiguration snmp
            contexts: ( Set with: (
                SNMPContextConfiguration snmp
                    name: SNMP.DefaultContextName;
                    values: ( Set with: 'SNMPv2-MIB' ) ) ) );
        securitySystem: ( SNMPSecuritySystemConfiguration snmp
            securityModels: ( Set
                    with: SNMPSecurityModelConfiguration snmpv0
                    with: SNMPSecurityModelConfiguration snmpv1 ) );
            adaptors: oacs;
            yourself
        ) new
```

As you can see, it is a bit more complex, and the creation method makes several assumptions about just how you want your engine configured, which, of course, you may change.

Engine Use

Engines are useful in themselves only as lightweigth SNMP clients. You can use an engine to send a message and get a response in two ways. The Opentalk SNMP Preview now supports an object-reference based usage style, as well as a lower-level API.

OR-Style Usage

If you play the object reference game, you get back an Association or a Dictionary of ASN.1 OIDs and the objects associated with them. For example, the port 3161 broker sets up its request using an object reference:

```
| broker3161 broker3162  oid ref return |

broker3161 := SNMPEngine newUdpAtPort: 3161.
broker3162 := self snmpv0CommandResponderAt: 3162.
broker3161 start.
broker3162 start.
oid := CanonicalAsn1OID symbol: #'sysDescr.0'.
ref := RemoteObject
      newOnOID: oid
      hostName: <aHostname>
      port: 3162
      requestBroker: broker3161.
^return := ref get.
```

This expression returns:

```
Asn1OBJECTIDENTIFIER(CanonicalAsn1OID(#'1.3.6.1.2.1.1.1.0'))->
      Asn1OCTETSTRING('VisualWorks®, Pre-Release 7 godot
      mar02.3 of March 20, 2002')
```

Object references with ASN.1 OIDs respond to get, set:, and so forth. These are translated into the corresponding SNMP PDU type, for example, a GetRequest and a SetRequest PDU in the two cases mentioned.

Explicit Style Usage

You can do the same thing more explicitly the following way, in which case you will get back a whole message:

```
| oid broker1 entity2 msg returnMsg |

oid := CanonicalAsn1OID symbol: #'1.3.6.1.2.1.1.1.0'.
broker1 := SNMPEngine newUdpAtPort: 161.
entity2 := self snmpv1CommandResponderAt: 162.
broker1 start.
entity2 start.
msg := SNMPAbstractMessage getRequest.
msg version: 1.
msg destTransportAddress: ( IPSocketAddress hostName: self
      localHostName port: 162 ).
msg pdu addPduBindingKey: ( Asn1OBJECTIDENTIFIER value: oid ).
returnMsg := broker1 send: msg.
```

which returns:

```
SNMPAbstractMessage:GetResponse[1]
```

Note that in this example, you must explicitly create a request with the appropriate PDU and explicitly add bindings to the message's binding list.

Entity Configuration

In the SNMPv3 architecture, an engine does not amount to much. It must be connected to several SNMP 'applications' in order to do useful work. And 'entity' is an engine conjoined with a set of applications. Applications are things like command generators, command responders, notification originators, and so on. There are several methods that create the usually useful kinds of SNMP entities, like

```
SNMP snmpv0CommandResponderAt: anInteger
```

Again, this invokes a method of greater complexity, but with a standard and easily modifiable pattern. There as several examples in the code.

MIBs

Opentalk SNMP comes with a small selection MIBS that define a subtree for Cincom-specific managed objects. So far, we only provide MIBs for reading or writing a few ObjectMemory and MemoryPolicy parameters. A set of standard MIBS is also provided. Note that MIBs are provided in both text and XML format. The Opentalk SNMP MIB parser required MIBS in XML format.

If you need to create an XML version of a MIB that is not provided, use the 'snmpdump' utility. It is a part of the 'libsmi' package produced by the Institute of Operating Systems and Computer Networks, TU Braunschweig. The package is available for download through http://www.ibr.cs.tu-bs.de/projects/libsmi/index.html, and at http://rpmfind.net.

Limitations

The Opentalk SNMP Preview is raw and has several limitations. Despite them, the current code allows a user, using the SNMPv2 protocol, to modify and examine a running VW image with a standard SNMP tool like ucd-snmp. However, one constraint should be especially noted.

Port 161 and the AGENTX MIB

SNMP is a protocol used for talking to devices, not applications, and by default SNMP uses a UDP socket at port 161. This means that in the absence of coordination between co-located SNMP agents, they will conflict over ownership of port 161. This problem is partially addressed by the AGENTX MIB, which specifies an SNMP inter-agent protocol. Opentalk SNMP does not yet support the AGENTX MIB. This means that an Opentalk SNMP agent for a VisualWorks application (only a virtual device) must either displace the host level SNMP agent on port 161, or run on some other port. Opentalk SNMP can run on any port, however many commercial SNMP management applications are hard-wired to communicate only on port 161. This places limitations on the extent to which existing SNMP management applications can now be used to manage VisualWorks images.

Opentalk

The Opentalk Beta is an extension of VisualWorks 7 and the Opentalk Communication Layer that provides tools for remote debugging, distributed profiling, load balancing, and communication using the SNMP protocol.

For installation and usage information, see the readme.txt file in the Opentalk beta directory.

SocratesEXDI and SocratesThapiEXDI

SocratesXML support at the EXDI level is included with this release in the **`preview/database/`** directory, in the SocratesEXDI and SocratesThapiEXDI parcels. The code is still under study and development for full release at a later time.

Currently this code supports:

- Supports MindSpeed 5.1 and SocratesXML 1.2.0 across Windows, Solaris and HPUX platforms.
- The SocratesXML API allows threaded calls, through thread safe drivers.
- All SocratesXML types (except MONETARY), collections and object references (OID) supported.
- Both placed and named input parameter binding is supported though SocratesXML only supports placed input binding.

Installation

SocratesXML 1.2.0

To install under Solaris and HPUX, simply load the SocratesEXDI parcel.

For Windows you must manually install the **`1880.016.map`** file. Do this by executing the external interface initialization code below and selecting the **`1880.016.map`** file:

```
SocratesInterface userInitialize
SocratesThapiInterface userInitialize
```

The class instance variable build defines the current build of the external interface classes on Windows platforms and can be ignored for the other platforms. The default value is set to 1881.016 on parcel loading.

MindSpeed 5.1

To install under Solaris and HPUX, simply load the SocratesEXDI parcel.

For Windows you must manually install the 1690.014.map file. Do this by executing the external interface initialization code below and selecting the 1690.014.map file when prompted:

```
SocratesInterface userInitialize
SocratesThapiInterface userInitialize
```

The class instance variable build defines the current build of the external interface classes on Windows platforms and can be ignored for the other platforms. The default value is set to '1881.016' on parcel loading.

Data Interchange

The Socrates database type to Smalltalk class mapping is given in table 1 below. Table 2 defines the mapping for database collection types.

The Socrates EXDI automatically converts Socrates database types to/from instances of concrete Smalltalk classes. Database bit types (BIT, VARBIT) are mapped to a new Smalltalk class BitArray. This class provides efficient uni-dimensional access to a collection of bits.

Table 1 - Socrates scalar type to Smalltalk class mappings

Socrates Data type	Smalltalk Class
BIT, VARBIT	BitArray
CHAR, NCHAR, VARCHAR (STRING), VARNCHAR	String
DATE	Date
DOUBLE	Double
FLOAT	Float
INTEGER, SHORT, SMALLINT	Integer
NULL	UndefinedObject

Table 1 - Socrates scalar type to Smalltalk class mappings

Socrates Data type	Smalltalk Class
NUMERIC	FixedPoint, LargeInteger
TIME	Time
TIMESTAMP	Timestamp

Table 2 - Socrates collection type to Smalltalk class mappings

Socrates Collection Data type	Smalltalk Collection Class
LIST, SEQUENCE	OrderedCollection
MULTISET	Array
SET	Set

Socrates support for heterogeneous collection maps naturally onto Smalltalk collections and is fully supported within in the limits defined by the SocratesXML C API.

For this release collections will be fetched and written in their entirety.

Reference Support

The Socrates EXDI provides transparent support for database object references, Socrates OIDs (similar to the SQL Ref data type).

A Socrates OID is represented by a lightweight Smalltalk object (class SocratesOID) that contains sufficient information to uniquely identify the database object across all accessible database servers. SocratesOID instances are not related to active database connections and so can exist outside the normal database server connection scope. SocratesOIDs can be instantiated back into live database objects (represented by instances of class SocratesObject) via an appropriate active connection i.e. one connected to the original database server.

Object Support

The Socrates EXDI provides access to raw Socrates database objects through instances of class SocratesObject. SocratesObject instances are intimately connected to the Socrates database server and so their scope is that of the underlying database connection.

A key feature of SocratesObject is high-level support for server side method (function) invocation. Simple server methods can be supported directly; methods with multiple or non-standard return values must be explicitly coded by the developer using in-build method invocation support methods. This typically involves defining a Smalltalk class (as a subclass of SocratesObject) to represent the target server class. This new class will be the place holder for both class and instance server method wrappers. All Smalltalk wrapper methods are defined as instance methods irrespective of whether they represent class or instance methods in the server. The Smalltalk wrapper methods are coded to extract the returned value(s) from the original method argument list, free any resources and returning the extracted value(s). The Smalltalk GLO hierarchy provides numerous examples of simple and complex wrapper methods.

GLOs

The Socrates EXDI supports LOB as a subset of the capabilities provided by SocratesXML GLOs. The Socrates EXDI implements the LOB interface through the SocratesGLO class hierarchy. SocratesGLOs provide a stream-like access to GLO data. All GLO subclasses have been modeled, i.e. audio, image and mm_root hierarchies. Each modeled subclass implements the majority of class and instance server side methods as Smalltalk methods. The user can easily add/extend this functionality by modeling any user-defined subclasses and server side methods.

The initial release of Socrates EXDI supports read-only support for Socrates GLOs.

Virtual Machine

IEEE floating point

The engine now supports IEEE floating-point primitives. The old system used IEEE floats, but would fail primitives that would have answered an IEEE **`Inf`** or **`NaN`** value. The new engine does likewise but can run in a mode where the primitives return **`Infs`** and **`NaNs`** rather than fail.

Again due to time constraints the system has not been changed to use this new scheme and we intend to move to it in the next release. In the interim, candidate code is provided as a goodie, and the engine can be put in the new mode by a **`-ieee`** command line option.

OE Profiler

The OEProfiler, an engine-level pc-sampling profiler now supports profiling native methods in the nmethod zone. The image-level code (**`goodies/parc/OEProfiler.pcl`**) is still only goodie quality but we hope to integrate properly these facilities with the Advanced Tools profilers soon.

User Settings

A new settings tool is being previewed in VW7. To install the new tool, load parcel **`preview/UserSettings/UserSettings.pcl`**.

The UserSettings Tool consists of a hierarchical list on the left, and a multi-function area to the right, following the model of the configuration tool found on most Linux desktops and the Netscape Navigator preferences tool. Selections in the hierarchical list on the left cause the associated configuration subcanvas to display on the right.

Each page appearing in the UserSettings Tool is defined using a method pragma. A typical page definition might look like this:

```
toolsWorkspace
    <settingsPage: #(tools workspace)
        label: 'Workspace'
        position: 0.019>
    ^WorkspacePreferences new
```

Browse methods in Tools.UserSettings, **setting pages** class method category, for the actual page methods.

The page layouts are defined by interface specifications in **ApplicationModel** subclasses.

Much of the current implementation is subject to change. This preview provides a first look at what is being developed for a future release.

Reader Comment Sheet

Name: ____________________

Job title/function: ____________________

Company name: ____________________

Address: ____________________

Telephone number: (   )   - ____________ Date: /   /

How often do you use this product? ☐ Daily ☐ Weekly ☐ Monthly ☐ Less

How long have you been using this product? ☐ Months ☐ Years

Can you find the information you need? ☐ Yes ☐ No

Please comment. ____________________

____

Is the information easy to understand? ☐ Yes ☐ No

Please comment. ____________________

____

Is the information adequate to perform your task? ☐ Yes ☐ No

Please comment. ____________________

____

General comment: ____________________

____

____

____

To respond, please fax to Larry Fasse at (513) 612-2000.

CINCOM®
The Smart Choice®