Cincom®

Cincom Smalltalk™ VisualWorks®

Release Notes

VisualWorks 8.2

P46-0106-24

SIMPLIFICATION THROUGH INNOVATION

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 8.2

Topics

These release notes outline the changes made in VisualWorks 8.2. 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.

Release notes for all actively-supported releases (i.e., support levels A-C) are included in the `/doc` directory of the VisualWorks installation. Notes for previous releases are also available, in the `/doc/ReleaseNotesArchive` directory.

For late-breaking information on VisualWorks, check the Cincom Smalltalk website.

Product Support

Support Status

Basic support policies for the current release are described in the licensing agreement. As a product ages, its support status changes. To find the support status for any version of VisualWorks and Object Studio, refer to the Cincom Smalltalk Support site.

ARs Resolved in this Release

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

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

Distribution Designations and Non-crypto distribution

In compliance with U.S. security requirements, VisualWorks now has a commercial version without encryption code.

Installation directories names are also being updated, as follows:

`<user-files>/vw8.2`

Full commercial distribution

`<user-files>/vw8.2pul`

Personal use license (non-commercial)

`<user-files>/vw8.2ne`

No encryption distribution

where `<user-files>` is the selected installation directory.

DLLs for Windows VMs

The set of DLLs required for the MS-Windows VMs (other than the static VM) have changed. On platforms older than Windows 10, these DLLs are not all guaranteed to be present by default, so the Installer will provide them if needed. For details, see the discussion of "MS-Windows DLL Requirements" in the *VisualWorks Installation Guide*.

Xtreams

Xtreams is a generalized stream/iterator framework providing simple, unified API for reading from different kinds of sources and writing into different kinds of destinations (Collections, Sockets, Files, Pipes, etc).

Xtreams is now included in the `/xtreams` directory of the standard distribution. The code in this directory is supported by Cincom, while the code that remains in `/contributed/xtreams` is not.

For instructions on usage, refer to the Xtreams documentation.

Known Limitations

While a large number of ARs (Action Requests) have been addressed in this release, a number remain outstanding.

Known Limitations sections are provided throughout this document, pertaining to specific product areas.

Purging Undeclared

When a parcel is only partially loaded, the purge of `Undeclared` will detect references in unloaded code. For example, if a loaded parcel named `MyParcel` contains the method:

```
FirstClassDefinedElsewhere>>extendingMethod
 ^SecondClassDefinedElsewhere
```

and both `FirstClassDefinedElsewhere` and `SecondClassDefinedElsewhere` are not loaded, then:

- extendingMethod will also not be loaded but it will be in the parcel's unloaded code
- Undeclared *will not* contain FirstClassDefinedElsewhere (unless it is referenced in some other place)
- If SecondClassDefinedElsewhere and all loaded references to it are removed, and it gets added to Undeclared, then a purge of Undeclared will fail to remove it because of the reference in extendingMethod, even though that method is not loaded and there are no references to it in loaded code

This issue has existed since unloaded code was introduced, is recognised as undesirable and will be addressed.

Limitations on Windows of displayShape: methods

On more recent versions of Windows 7 it is not permitted to draw directly on the screen. Various methods in the Screen class in VisualWorks attempt to do so. This is implemented by creating an invisible window, drawing on it, and then destroying the window. However, this is a slow process, and methods that attempt to do animation using this primitive can become very slow and the animations can become effectively invisible. Use of these mechanisms is often used for animation of operations like dragging within a graphical editor. In such cases they can be replaced by graphics operations within that window, which will be much, much faster.

Operations like Rectangle>>fromUser: are not as slow because they can create the invisible window once, do numerous drawing operations, and then complete. But methods like the Screen>>displayShape:... family do not have this information. At the moment, use of these methods is discouraged while we examine possible solutions.

Publishing a Bundle as a Parcel

When **Publish as Parcel** is invoked on a bundle, we recommend selecting **Include bundle structure.**

If you choose not to select this, check that the bundle's own prereqs are what the parcel will need, since the prereqs of its subpundles will then not be assigned to the parcel (which may therefore show unexpected behaviour on load).

In a subsequent release, the bundle structure will aways be saved when saving a bundle, and will no longer be an option.

AppeX ServerMonitor needs manual refresh in IE

Late testing has uncovered a bug in the AppeX ServerMonitor: when it is first opened in Internet Explorer, the ServerMonitor page may not display the list of responders for the expanded Server.

The page can be repeatedly refreshed until the responders for the server are displayed. This may take several refreshes. In order for refreshing to work, the server must be the first in the list of servers in the ServerMonitor. This problem is currently not observed in Firefox or Chrome.

JSFile script directory needs manual initialization

Late testing revealed a problem in initializing the default script directory for the JSFile framework. (For documentation, see the "Javascript" chapter in the *Web Application Developer's Guide*.)

Switching to **Supply Javascript code from: Script files** (either on the **Web Development > JavaScript** page of the Settings Tool, or by evaluating `Application codeComposer: JSFileComposer`), before explicitly setting the script directory provokes an unhandled exception (UHE), and the Smalltalk image will not switch to using script files.

In order to avoid this problem, before switching to script files, initialize the scripts directory by taking one of the following two actions:

1. Open the Settings Tool, navigate to the **Web Development** page, select a directory in the **Script Directory** field, and click **Apply**, or:
2. Evaluate the following in a workspace:

```
Application scriptDirectory.
```

If either of these actions is taken *after* the UHE is encountered, then the setting to use **Script files** must be toggled off and on again before the image will start using script files.

Chapter

2

New and Enhanced Features

Topics

This section describes the major changes in this release.

Base Image

Correct resolution of Fractional Seconds

Fractional seconds that were not processed to the currently available resolution (nanosecond) in various VisualWorks Time objects have been corrected.

Support for Short Compiled Methods has been Removed

In VisualWorks 8.2, the last remaining support for short compiled methods has been removed. Short compiled code was an optimization that allowed encoding bytecodes of a method (or a block) as one or two SmallIntegers instead of a normal ByteArray instance, assuming the entire body of the bytecodes would fit into those. This was removed in VisualWorks 7.7, with final clean up in the current release to remove loose ends. Any short compiled code stored in parcels (or binary-published versions in Store) is converted upon load.

Database

General Enhancements to the Database Framework

As in VisualWorks 8.1.1, the Sybase (CTLibEXDI) interface supports 1/300th of a second precision for DATETIME and TIME types. Passing bound arrays holding FixedPoints values works in ODBC. In VisualWorks 8.2, the shutdown of an image with a stale connection to MySQL is cleaner. Loss of a connection to Postgres is handled better, as well.

Enhanced Support for Scrollable Cursors

VisualWorks 8.2 includes changes to increase tolerance for out-of-range scrollable cursors. Cursors can now either move out of their data ranges from top or bottom, and they will still be able to work correctly. The Oracle, ODBC, DB2 and Sybase connects all support this feature.

Enhancements to the PostgreSQL EXDI

In previous releases of VisualWorks, if multiple image threads were used to run sessions on the same PostgreSQL socket connection, it was still possible to evade the protective buffering of results if the clash hit an exact moment in the code. This has been fixed. An effect of the fix is that a session writing to the database will delay a threaded session on the same connection that seeks to read from it, and vice versa.

Installing a module into a PostgreSQL installation can generate a type whose ids are not the same as that of the same type created in another installation. (An example is the `HSTORE` type that accompanies `JSON` and `JSONB`.) Thus, the type's id cannot be hardcoded in our interface. Our PostgreSQL 3.0 driver now lazily gets the ids for such types if user code needs them.

On Mac OS X, the PostgreSQL 64-bit interface is no longer mapped to the 32-bit interface; it now uses a 64-bit installation.

PostgreSQL always provided some server and connection parameter data on login: the `get*ConnectionParameter*` methods now simplify getting non-login parameters.

Enhancements to the SQLite EXDI

Class `SQLite3Interface` has hard-coded values for the `libraryDirectories` attribute, which specifies all the candidate directories expected to contain your SQLite library file (e.g., `.dll` on Windows, `.so` on Linux, etc.). VisualWorks searches only these candidate directories or, if no candidate directories are specified, it searches in the system path (e.g. `PATH` on Windows, and `LD_LIBRARY_PATH` on Linux).

If your SQLite library is not located in the list of `libraryDirectories`, specified in class `SQLite3Interface`, you have two options.

First, you can add your SQLite directory to this list, using the existing pattern. Or, second, if the library is somewhere in the system path, you can simply modify the class definition, removing the candidate entries corresponding to your platform.

For example, on Windows, remove all the `[win]` entries from the class definition for `SQLite3Interface` and save it (that is, pick **Accept** from the

<Operate> menu). Once this is done, the virtual machine will use your system path to find and load the library file.

Also in this release, SQLite now reports server-configurable runtime limit settings, and raises errors for out-of-range integers.

Enhancements to the Oracle EXDI

In this release, the Oracle EXDI supports binding of arrays and binding by position in Oracle PLSQL (previously, only variables could be used).

Both `OracleEXDI` and `OracleThapiEXDI` now function properly on 64-bit OS X.

The following notes may help when configuring OS X to use Oracle's Instant Client with VisualWorks.

Since Oracle distributes their 32-bit and 64-bit interfaces in two separate libraries, you cannot have your machine configured to use these two libraries at the same time.

1. First download the Oracle Instant Client from their website and install both the 32-bit and 64-bit version of the following Instant Client packages (in separate locations):
   - Basic (or Basic Lite)
   - SQL*Plus
2. Follow the special installation instructions at the bottom of the download page, specifically create the appropriate symlinks in *each* installed version of the Instant Client. For example:

   ```
   cd ~/oracle/instantclient[32|64]_11_2
   ln -s libclntsh.dylib.11.1 libclntsh.dylib
   ln -s libocci.dylib.11.1 libocci.dylib
   ```
3. If you launch VisualWorks in a shell, you will need to update your **`.bash_profile`** (or whatever you are using) to add the following environment variable. Note that you can only specify the specific version (where NN=32|64) of the Instant Client you wish to use at any one time.

   ```
   DYLD_LIBRARY_PATH=~/oracle/instantclientNN_11_2_0_4
   ```

4. If you launch VisualWorks via double-click (not really advisable) or by drag/drop, you will need to read the information here:

   ```
   http://stackoverflow.com/questions/25385934/setting-environment-variables-via-launchd-conf-no-longer-works-in-os-x-yosemite
   ```

   and create your own startup plist to provide this environment variable to the launchd environment in which we run VisualWorks.

Our testing has not yielded positive results with Oracle's suggestion to put symlinks into the `~/lib/` directory. This strategy works when running Oracle's SQLPlus or VisualWorks from the command line without setting the `DYLD_LIBRARY_PATH`. but it seems that VisualWorks in the launched environment requires the presence of the `DYLD_LIBRARY_PATH` environment variable. Others have also recommended setting the `DYLD_FALLBACK_LIBRARY_PATH` as well.

Remember that any change in the `.bash_profile` or startup plist will require either rebooting your machine, or running something that will get the new environment setup.

Finally, it's worth noting two things about the startup plist approach:

1. Any changes you make to your `$PATH` envionment variable in your plist will not be seen in VisualWorks.
2. Apple is in the process of deprecating the `DYLD_LIBRARY_PATH` environment variable, and though it still works (in Mavericks, Yosemite, El Capitan) a different approach may be needed in the future.

DLLCC

Split the Hierarchy of ObjectiveCRuntime into 32- and 64-bit

The shared variable `ObjectiveCRuntime` is now an instance of either `ObjectiveCRuntime32` or `ObjectiveCRuntime64`. DLLCC does not adjust structures based on inherited types in a class hierarchy, which became a problem for 64-bit Objective-C code, where the concept of a Float type changed from 32-bit to 64-bit while using the same type name. To avoid this problem, use the methods #pointType, #sizeType, #rectangleType, etc., instead of using the types directly that contain floating point data.

Glorp

Enhancements to the Glorp framework

In the VisualWorks 8.2 release, Glorp now writes groups of row updates in a single round-trip. Previously, the same number of rows could be inserted quickly but updated slowly as the latter was being done in single trips to the database.

Group-writing of inserts can be done across a much wider range of platforms and uses a more succinct protocol. Grouped writing can also be done when rows have a column whose values are server-assigned. (Previously, this always had to be done in single steps. Now, Glorp recognises when the value assigned to one row is being referenced in another row in the same batch, and only writes single rows in that case.) More platforms can bind arrays of values to write multiple rows. These writing speed improvements affect the following and later versions: Oracle 9.0, SQLServer 10.0, DB2 8.0, SQLite 3.7.11, MySQL 5.0 and Postgres 8.2.

In previous releases, the dual use of the column default value in Glorp could cause confusion. Fields now have a separate dbColumnDefault and imColumnDefault. The former configures the database server to provide a default value for an unset/NULL value in an insert row. The latter configures Glorp to do that in the image before writing the row to the database.

In this release, arithmetic on dates and timestamps bound into SQL is handled better. String matching between main and subselects (queries within queries) works better. Glorp meta-data and migration acquires scale and precision of numeric types more reliably.

The "where clause" of a query can be edited by AND:ing or OR:ing expressions (from an existing query or, for power-users, constructed by hand), then either creating a new query via

#read:where: or resetting the where clause on the existing query. In previous releases, the latter was settable by indentical methods #where: and #whereClause:. The former method is now deprecated; use #whereClause: to reset the value on the instance side.

TimedExpiryCachePolicy is inconsistent with VWDatabaseAccessor>>reusePreparedStatements

Glorp's reusable statement cache has been optimised. Any users who have overridden the method:

```
VWDatabaseAccessor>>initializePreparedStatementCache
```

must adapt their code to its changes.

GUI

GUI/Tools Upgrade FAQ

The VisualWorks 8.0 release includes a significant upgrade to implement new platform-faithful UI Look and Feel (UISkinning) and major improvements to our widget layout processing (UILayout). These changes have impacted our existing menu processing and many of our current widgets in a variety of ways. The following topics identify issues which will require attention in your applications as you upgrade to this release.

Replace sends of #pressAction with #trigger

With the introduction of class ViewEventController as the first step towards improving our widget event processing, the #pressAction method of various button controllers has been deprecated in favor of the more universally implemented #trigger which allows either the view or the controller to respond to the click event as appropriate.

The reasoning behind this is that the selector #pressAction describes a particular UI technique (pressing), rather than describing the desired effect (triggering).

Specifying a layout using a Rectangle or a Point

This potentially affects sends of #layout:, #layoutFor:, #newLayout:, and others. Convert a Rectangle or Point to a Layout instance by sending #asLayout to it.

Using a simple Rectangle or Point as a layout is no longer acceptable with the new UILayout mechanisms. The SpecWrapper requires a LayoutFrame instead, which can be acquired by sending #asLayout to a Rectangle or Point.

Deprecation notices for MenuItem >> command:

We are in the process of rolling out menus using the new Commands framework, but that is a work in progress. The original method continues to work, and many of our menus have not yet been upgraded, so this currently produces too much Transcript noise. Work around this by commenting out the deprecation notification line in the method MenuItem>>command:.

ComboBox widget revisions

Replace references of ComboBoxView with ComboBoxInputFieldView.The ComboBox widget had been significantly restructured in order to, among other things, make its drop-down menu behave according to platform expectations. Classes have been renamed and accessing internal components of this composite widget have changed. Please review any extensions you have made to this widget or any code that references the internal components.

VisualWave UI construction on a headless server

The changes for UISkinning have impacted VisualWave's ability to appropriately identify a row/column position for certain widgets (most notably text labels) when creating HTML tables to generate a web layout that approximates that of the native VisualWorks window. Since this issue only shows up with a headless image, we expect this is due to the absence of a reference font description which can be used to determine a widget's bounds in relation to its neighbors.

The workaround is to modify any widgets in the UIPainter so that any text (especially for labels) uses the HTML default font instead of relying on the System (Widget Text). Unfortunately this is not easily automated, since it requires adding a new line to the spec to specify the font for the widgets in question, as opposed to changing an existing parameter in a spec.

Progress bar widgets don't follow user color settings or show the % as text

The old look isn't platform compliant, and doesn't correspond to any native look, and hence is no longer available. Currently, you need to do a custom skin to get that appearance. We are looking at adding a variant look mechanism to the skins make this easier (AR 71321) and specifically providing the old progress bar look as a variant (AR 71322).

Remove InputState>>screen

In this release, all uses of `InputState>>screen` have been replaced by `Screen>>default`.

VisualLauncher widgetAt: #statusBar returns an empty CompositePart instead of the statusBar

`WidgetWrapper>>widget:` has been removed. Most uses of this method would reference `SpecWrapper`, which is a subclass of `WidgetWrapper`. The wrapper now uses `component bottomComponent`, which should always be the widget, so you don't need to use `widget:` to set the widget. Set the `component` instead, or just let it return the `bottomComponent`.

Pixmap and Mask allocation fail for non-integer extents

When creating a `Pixmap` from the measurement of a string, on OS X the measurement can be floating point which will cause the `Pixmap` allocation to fail. VisualWorks currently contains a lot of code that assumes the relationship is always integer, but that's not the case anymore. To avoid these allocation errors we have made `Pixmap` (and `Mask`) more flexible with width and height so that:

```
Pixmap extent: 53.395 @ 14
```

creates a Pixmap with width = 54 and height = 14, giving it an extent of 54 @ 14.

When calculating spacing for layout you should not rely on the original extent provided to Pixmap, but get the resulting extent from the newly created Pixmap.

Consistent Behavior for #selectOnDownIfDragEnabled

We have updated the default for #selectOnDownIfDragEnabled in SequenceViewSpec to match the default value set in SequenceController for #selectOnDownWithDrag. This default is now consistent for a SequenceView both in its Spec and its Controller. The default behavior will now be that attempting to click and drag an unselected item in such a view will no longer automatically select the item before the drag operation starts. This is standard behavior consistent across platforms, and now enables the a view to permit dragging multiple selections. Prior to this change, the click to initiate the drag operation would unselect all items other than the one at the mouse click location.

This default can be overridden in the UIPainter for any SequenceView or subclass.

This change may require that you update existing SequenceViews in your UIs if you have not protected the method referenced as the #dragStartSelector from initiating a drag operation when there is no selection in the view. The protection can either be implemented in your #dragOkSelector method (preferred), or a test can be added into your #dragStartSelector method.

Internationalization

Support for PerProcessCatalogs integrated in the Base system

The functionality in the PerProcessCatalogs component has been moved into the Base system and as such is now always available without needing to load anything additional. Any prerequisite in customer code that references PerProcessCatalogs may be removed. The PerProcessCatalogs parcel has been moved to `/obsolete` on the distribution ISO as of this release. It is now an empty parcel containing only prerequisites, to satisfy those who build against it.

New Convenience Methods

Convenience methods for using the #custom print policy format have been added and documented.

Updated Class Comments

The class comments for TimestampPrintPolicy and TimestampReader have been updated.

Net Clients

Support for Bearer Authorization Tokens

VisualWorks 8.2 includes support for Bearer authorization tokens, as defined in RFC 6750.

To create an authorization field with a Bearer token, use the following methods in class AuthorizationField:

bearerToken: aString

The String argument will be encoded using base64 encoder.

bearerB64Token: aString

The String argument should be base64-encoded.

Example:

```
request := HttpRequest
    get: 'https://www.example.com/oauthExample/'.
request
    addField: (AuthorizationField bearerToken: 'mF_9 .B5f-%4.1JqM').
```

The request will include a base64-encoded String:

```
Authorization: Bearer bUZfOSAuQjVmLSU0LjFKcU0=
```

LDAP

VisualWorks 8.2 now includes support for LDAP. For this, the unsupported LDAP package has been moved from `/contributed/LDAP` to the `/net` directory, and the LDAP classes were refactored to allow expanding them with a TLS connection. The `/contributed/LDAP` directory now includes only the `LDAPTestUI` parcel, which remains unsupported.

We also added the LDAPS component, which implements LDAP over TLS secure extensions, following the specification in RFC 4513. This code has been released as the `LDAPS` parcel in the `/net` directory.

LDAPS has been implemented with two classes:

LDAPSConnection

Implements the StartTLS operation and allows using LDAP over TLS. By default, the implementation uses the StartTLS operation to establish a secure connection. It is also possible to start the non-standard LDAPS ("Secure LDAP", commonly known as "LDAP over SSL") protocol on a separate port, which is set to 636 by default.

LDAPSConnThread

Opens a TLS connection.

To use a StartTLS operation via port 389:

```
connection := LDAPSConnection new
      tlsSubjectVerifier: [:cert :conn | true];
      yourself.
"Establish a regular connection"
connection connectToHost: 'localhost' port: 389.
["Apply the TLS layer"
 connection performStartTLS.
] on: Xtreams.TLSCertificateWarning do: [:ex | ex resume. nil].
"Authenticate"
connection
 authenticateAs: 'dc=mydc'
 password: 'mypassword'
 constraints: connection defaultConstraints.
```

or:

```
connection := LDAPSConnection new
      tlsSubjectVerifier: [:cert :conn | true];
      yourself.
"Establish a regular connection"
connection connectToHost: 'localhost' port: 389.
```

```
["Apply the TLS layer and authenticate. By default the #startTLS operation is performed
 before authenticating the credentials"
 connection
  authenticateAs: 'dc=mydc'
  password: 'mypassword'
  constraints: connection defaultConstraints.
] on: Xtreams.TLSCertificateWarning do: [:ex | ex resume. nil].
```

To use LDAPS (LDAP over TLS, via port 636):

```
connection := LDAPSConnection new
      tlsSubjectVerifier: [:cert :conn | true];
      "establish a TLS connection as soon as
      the client connects to the server"
      useLDAPS: true;
      yourself.
[connection
 connectToHost: 'localhost'
 port: 636
 dn:  'dc=mydc'
 password: 'mypassword'.
] on: Xtreams.TLSCertificateWarning do: [:ex | ex resume. nil].
```

Tools

Autocomplete

VisualWorks 8.2 includes a new autocomplete feature, enhancing the existing autocomplete that was introduced in VisualWorks 8.0.

The VisualWorks 8.0 autocomplete would attempt to predict what you were typing and type it out for you. It would select the completed text for you, to allow you to change course and type something different. You could also press escape to reject its suggestion. This approach to autocomplete would only offer you one suggestion at a time and the information gleaned by a developer was limited.

In VisualWorks 8.2 we have introduced a new autocomplete that acts as a popup as you code. It offers as many suggestions as it thinks might be useful and allows keyboard navigation through the potential options. Its suggestion capabilities have been significantly increased too.

The autocomplete can now suggest pragmas, getters and setters, suffixes to multi-argument keywords, or the method name you're currently implementing, if you're creating a recursive method call. It can search the hierarchy when suggesting a selector sent to super and can perform a few other tricks as well.

What the autocomplete can find is extensible by adding new pragmas to `SmalltalkAutocompleteSearch`, as well as how to present that information found in `SmalltalkAutocompleteModel`. What to do when the correct suggestion is chosen can also be specified in `SmalltalkAutocompleteModel`.

The autocomplete can be used from code editors, the debugger and the workspace if it is in **Style as Smalltalk** mode.

Inline modal dialogs

When an error or warning is displayed in the VisualWorks source code editor, it will often contain potential actions a developer could take. In VisualWorks 7.x and older releases, these errors and warnings were displayed sequentially using modal dialogs.

A significant drawback of using dialogs was that it often took several keyboard operations to choose an action, especially when trying to correct something misspelt.

This was changed in VisualWorks 8.0 and 8.1 to show all the warnings and errors at once inline in the source code. However, there was no way to invoke these actions via shortcut keys, as you could in the dialogs of 7.x.

To improve on both approaches, the source code editor in VisualWorks 8.2 now provides inline modal dialogs. The modality depends on the position of the input cursor. E.g., if there is an action to add a temporary variable, the **T** key is bound as the shortcut key to perform that action while-ever the input cursor is within the source code element that the warning or error applies to.

The shortcut keys are typically linked to the first letter of the warning or error and spelling corrections are bound to the numeric keys 1..9. A double underline is placed inside the warning or error to indicate what the shortcut key is and when the input cursor leaves that focus, the double underline disappears too. This allows you to change the focus with regular keyboard navigation to correct a different warning or error instead of the initially-focused one.

Browser Navigation history with backward/forward buttons

In VisualWorks 8.2, individual System Browser windows now have **Backward** and **Forward** toolbar items to quickly navigate to previous code components, whether they be pundles, classes, categories or methods.

This information is pooled in the **History** menu that is shared across all Code Browser windows. Pundles, classes and methods visited the most often are shown in this menu as a quick way to jump back to something that may be far away in the back/forward history of the window, or came from the back/forward history of a different window.

Make Command/Control+E find implementors of a symbol when performed on a Symbol in source code

Previously, the VisualWorks tools included an **Explain it** feature that would tell you what you were clicking on and provide an option to browse its class. In VisualWorks 8.x, the source code editor can now perform an analysis of what you are clicking on, which eliminates the need for a dialog that asks the developer to choose. In VisualWorks 8.2, this analysis feature has been extended to adjust more menus than in the previous releases. The **Explain it** hot key control+e was replaced by **Explore** and a second hot key was added, control+shift+e to find references to the thing you might otherwise explore.

Generalise the #languages information in the Source Code Editor

There is an on-going effort to enhance the code editing and refactoring features of the System Browser such that they provide greater support for languages other than Smalltalk. At present (release 8.2), VisualWorks actively supports the editing of both

VisualWorks and ObjectStudio Smalltalk, as well as JavaScript for AppeX.

Previously, the Browser was hard-coded to support only one dialect of Smalltalk, using class RBParser. Some features of the Browser still invoke RBParser, but over time those will be removed too. A registry of coding languages now exists on the class LanguageServices which provides a concrete set of knowledge per-language.

The compiler, parser, some lexical rules, and plugins for the code editor are provided on a per-language basis, and the kinds of information stored on LanguageServices can be extended too.

The language can be set on a SourceCodeEditor, otherwise it will be implied by the receiver class of the code it's editing. If there is no receiver class available it will assume Smalltalk. This conforms with the practice of organising code into classes and having those classes have a different set of services available in the development environment.

If the parser of a non-Smalltalk language produces an RBMethodNode parse tree, some existing refactorings will work without any changes required, such as renaming a class or selector referenced in that source code.

Refactor InspectorShell to allow embedding in other window specs

In previous releases of VisualWorks, class InspectorShell attempted to act as an embeddable inspector from Trippy but it was so strictly tied to the idea of being in charge of the window that it never worked that way. In release 8.2, InspectorShell has been split into two classes, InspectorShell and InspectorWindow.

Class InspectorShell attempts to control the menu and toolbar if the menu and toolbar match what it expects from an InspectorWindow, otherwise it quietly allows the operation of inspecting objects to work, with tabs, while being embedded in another window.

Class InspectorWindow continues to behave as InspectorShell used to behave, i.e., it expects to own and control the whole window.

Add shortcut key to create temp from undefined

Before the Source Code Editor (SCE) was modernized, compiling a method could present you with dialog after dialog. Those dialogs could let you correct things by the keyboard - tab, space, enter, shift +tab, picking some things from lists, etc. It wasn't beautiful, but it was practical.

With the SCE we dropped these modal dialogs and inserted the compiler warnings and errors in to the source code in a non-destructive manner. This allows you to ignore or skip or resolve in any order you chose, the various issues in the source code, but it was mouse driven only.

In VisualWorks 8.2, the SCE now includes what we're calling "inline modal dialogs". When the input cursor is in a warning or an error, specific actions may have shortcut keys that can be used. Those keys are owned by the inline modal dialog while-ever it has input cursor "focus", but you can navigate away with the cursor keys or with the mouse at any time. The shortcuts won't be owned by that particular inline modal dialog and it will visually remove the indications of which keys are bound to it.

This gives you the best of both worlds: a non-dialog interrupting save/compile experience with the power of shortcut keys to resolve common issues.

Ctrl-B is duplicately bound in Source Code Editor

The control keys for adding/removing a breakpoint in the SCE have changed to (Ctrl-Cmd)+/ to avoid conflicting with the historical use of (Ctrl-Cmd)+B which remains bound to **Debug it**.

Provide a way to skip the 50ms delay when writing to the Transcript

The `Transcript` bunches writes together and commits them to the Visual Launcher when its internal buffer is full or 50ms has elapsed. To guarantee writes to the `Transcript` are committed, you can perform a `Transcript flush`. This will instantly flush the internal buffer to the Visual Launcher. Be careful not to commit too often if there are lots of writes being sent to the `Transcript`, otherwise the point of the internal buffer will be defeated. Consider either not using `#flush` or if necessary, doing so at the end of a full line of log details.

Browsing or Renaming Classes and Shared Variables no longer includes Symbols

In VisualWorks 8.1.1 and earlier, searching for or renaming a class or shared variable would also find or rename symbols that matched the object's name and were in its scope (i.e. could see it unqualified, so could be references to it).

For example, let’s say a developer defines a database connection subclass `OracleConnectionForMyApp`, and has profiles for it hard-coded (in code that is in-scope of the class), e.g.:

```
myAppProfile
  ^ConnectionProfile new
        driverClassName: #OracleConnectionForMyApp;
        ...;
        yourself
```

In this situation, searching for `OracleConnectionForMyApp` also finds the `myAppProfile` method, and renaming the class also renames the symbol in that method. This has been corrected in VisualWorks 8.2.

Virtual Machine

MS-Windows VMs built using Visual Studio 2015

MS-Windows VMs are no longer built with VisualStudio 2010, but now are built with Visual Studio 2015.

WWW

AppeX Scaffolding Tools

VisualWorks 8.2 includes new AppeX Scaffolding Tools for creating web applications from existing database tables. This was in preview for VisualWorks 8.1, and has now been integrated into the release proper. The tools may be used by loading the `AppeX-Scaffolding-`

Tool and AppeX-Examples-Scaffolding packages. For details, consult the "Scaffolding" chapter of the *Web Application Developer's Guide*.

JS-File support in AppeX

In VisualWorks 8.2, the new JSFile framework enables developers to write out AppeX JavaScript code to the file system, change it there, and automatically reload it in the Smalltalk image, reparsing it as necessary.

JSFile support may be used, for example, to facilitate RAD development of JavaScript in AppeX, in conjunction with the Chrome development tools.

Minification support in AppeX

The Minification framework enables developers to generate minified code for JavascriptLibraries.

This minified code is generated during the development process, and cached for each JavascriptLibrary, so that it can be transparently delivered to the client upon request for a JavaScript resource at runtime. To use this minified JavaScript in production, deliver the production Smalltalk image with the minified code pre-cached.

Support for Internationalization in AppeX

VisualWorks 8.2 includes AppeX-Internationalization, a new component that extends AppeX with a client side JavaScript/HTML5 Internationalization/Localization framework.

In place of hardcoded display strings in JavaScript code, the developer substitutes a _translate method call with a key and a default value as parameters. At runtime, the web client downloads the locale (e.g. English) catalogs, and when executed in the web client, the _translate function looks up the appropriate display value for each key provided.

Contributed

Third-party Components

The VisualWorks distribution contains Roassal2 (in the directory **/contributed/Spy**). For details, consult its package comment. (It also still has Spy and its prerq Roassal — i.e. Roassal1 — in the same directory; do not load both into the same image.)

Some goodies have been removed from the distribution, but they can be found in the Cincom Public Repository, or at the Cincom Smalltalk website on the Contributed Components page.

Documentation

Application Developer's Guide

Updated for the current release. Updated discussions of the source code editor, syntax-highlighting, and the new auto-completion features. Fixed a problem screen shot in the chapter on name spaces.

Database Application Developer's Guide

Document support for JSON in ODBC, including numerous code examples. Enhance documentation for cursors and scrollable cursors in Oracle, ODBC, DB2 and Sybase. Document support for Oracle on 64-bit OS X. Amend SQLite EXDI documentation to mention library-not-found workaround. Document support for PostgreSQL's NoSQL features.

DotNETConnect User's Guide

Updated for the current release.

Glorp Guide

New chapter added to cover use of the Glorp Atlas. Enhance discussion of mapping Dictionaries. Enhance discussion of Active

Record. Amend discussion of Glorp migration. Minor editorial corrections.

Installation Guide

Updated for the current release. MS-Windows DLL requirements revised, updated. Converted document to use the new template.

Internationalization Guide

Documented the codes used to specify print policy formats for `TimestampPrintPolicy` and `TimestampReader`. Chapter 2 of the Guide has been updated with details of the revised codes in use since VisualWorks 7.7. Documented new convenience methods for using the `#custom` print policy format.

Net Clients Developer's Guide

Updated for the current release. Enhance documentation of secure SMTP. Document support for `HttpRequests` with non-UTF-8 form data. Enhance discussion of class `SimpleBody`. Clarify usage of the Logging Tool. Document support for Bearer authorization tokens per RFC 6750. Correct several erroneous code examples.

Tools Guide

Updated for the current release. Document updates to code styling and auto-complete features.

Web Application Developer's Guide

General update for the current release to cover new functionality. Editorial cleanup. Document the AppeX Scaffolding framework, JSFile, minification, and internationalization support. Document Create Web Application Tool and corresponding API.

Web Server Developer's Guide

General update for the current release to cover new functionality. Updated discussion of WebSockets to cover changes in the SiouX framework. Amend discussion of SiouX server configuration and deployment.

Reorganize Release Notes

With VisualWorks 8.2, the Release Notes have been reorganized. Previously, all Release Notes were included in the `/doc` subdirectory. From 8.2 forward, only the notes for actively-supported releases (i.e., support levels A-C) are included in the `/doc` directory. Release Notes for previous versions of the product are still available, but have been moved to the `/doc/ReleaseNotesArchive` directory.

Added list of legacy frameworks to the Release Notes

The VisualWorks Release Notes now include a list of legacy frameworks. Legacy frameworks have minimal to no support and their functionality will not be enhanced going forward. We keep some of these frameworks available in the product, as some customers may want to use them longer, need extra time to port to a newer technology, or may wish to continue with and maintain the framework for themselves.

New Documentation Template

In this release, a new template for the PDF documentation makes its first appearance. For the moment, only a few documents make use of it (e.g., these *Release Notes*), but eventually all VisualWorks documentation will be converted.

Chapter 3

Preview Components

Topics

Several features are included in a `/preview` and available on a “beta test,” or even pre-beta test, basis, allowing us to provide early access to forthcoming features. Several are described in the following sections. Browse the directory contents for last minute inclusions.

Glorp

Recursive Queries

VisualWorks 8.2 introduces a very limited ability for Glorp to perform recursive queries: i.e., using the result of one query as data for the next repeatedly within a single round trip to the database. This feature will become more robust in VisualWorks 8.2.1 and will be supported in 8.3.

For example, in a Store Workbook, you can execute:

```
| query childId |
childId := 529.    "the id of some bundle whose ancestors are to be found"
"Define the start point"
query := (Query
    read: StoreBundle
    where: [:each | each primaryKey = childId])
  retrieveAll: (Array    "and what to retrieve from it"
      with: [:each | each primaryKey]
      with: [:each | each trace primaryKey])
  thenFollow: #trace        "the mapping whose closure is wanted"
  recursivelyRetrievingAll:   "retrievals that will populate the recursion"
    (Array
      with: [:each | each recurse primaryKey]
      with: [:each | each recurse trace primaryKey])
  intersect: (Query read: StoreBundle).    "return this joined to the recursion result set"
session execute: query.
```

This returns the closure of the #trace mapping for the chosen StoreBundle, i.e. all its ancestors. (For additional examples, see class GlorpRecursionTest in parcel GlorpTest.)

GUI

Upgrading an Existing Application using Classic Skins

VisualWorks 8.2 provides a separately-loadable parcel in preview, UISkinning-Skins-Classic, which implements a set of skins corresponding to the old emulated platform look policies, but with a more modern look. These skins are usable either on their native platform or in an emulated form on a different platform. They have been developed to honor the text attributes and font expectations of applications developed with pre-skinning versions of VisualWorks.

This is beta-level preview code, and as such it's still a bit fragile on the edges. In addition, the work currently underway to provide layout compatibility is incomplete and not available yet for use with this package.

How to use the VisualWorks Classic skins:

1. Load the `UISkinning-Skins-Classic` preview parcel.
2. Select the desired classic skin in the Settings Tool.

Some caveats and known issues with the current code:

- The skin must be set at the image level. It is not possible to apply a skin to an individual UI.
- The package implements the default (system) fonts and text attributes. It may or may not work for Small, Large, Fixed, Named fonts, etc in widget or UI definitions.
- The default font size for Windows classic skins on Mac OS X is too small.
- There are still a number of anomalies related to layout between these skins and their 7.x counterparts. Most notably the `SpinButton` on Windows is still not as we desire, and various vertical positioning is sometimes misaligned.
- The Browser is not able to refresh the `Artists` (which hold their skin) for the Navigator's `TabControlBarViews`. This means that when switching between standard and classic skins, you will not see the full effect of such a change, especially in the Browser lists, until you open a new Browser. Menus are also similarly affected.
- You *must not* try to unload the package with your image using one of the classic skins. In addition, after you switch back to a Native or Default skin prior to unloading, you *must* close and/ or re-open all Browsers before you unload. If you forget these warnings, your image will go into a GC trance and crash.

We welcome feedback about whether this first step in backward compatibility looks promising to customers upgrading from 7.x versions of VisualWorks. Our current plans are to continue to improve the font and text attribute selection and add the layout changes as soon as they are ready. Once that is working well, we will also be looking into adding things like widget and/or window foreground and background colors defined in the UIPainter. In addition, we are investigating how to add the ability to specify a skin at the `ScheduledWindow` level. If there are other UI-related compatibility issues we need to look at, we would appreciate hearing about them as well.

Virtual Machine

Users who have interest in any of these new high-resolution (retina) display or 64-bit support on certain platforms are encouraged to use the preview VMs and provide feedback.

Preview Virtual Machines for OS X

VisualWorks 8.2 includes a preview VM for 32-bit VisualWorks on OS X / MacOS that may provide better graphics on retina displays than the production VM, but may have some remaining issues. It can be found here:

```
$VISUALWORKS/preview/bin/macx
```

We have added a preview VM for 64-bit VisualWorks on OS X / MacOS that will run 64-bit images (which was not available on OS X until now), and also includes the retina graphics support of the 32-bit preview. It can be found here:

```
$VISUALWORKS/preview/bin/macx64
```

Preview Virtual Machine for 64-Bit Linux on PPC

We have added a preview OE or 64-bit VisualWorks on Linux PowerPC. It can be found here:

```
$VISUALWORKS/preview/bin/linuxPPC64
```

Universal Start Up Script

Unix-based Systems

This release includes a preview of new VisualWorks loader that runs on all Unix and Linux platforms. This loader selects the correct object engine for an image, based on the image version stamp. Formerly, the only loader of this sort was for Windows.

The new loader consists of two files and a readme in `/preview/bin`. Installation and configuration information is provided in the readme.

This loader is written as a standard shell script which allows it to be used to launch VisualWorks on virtually any Unix-based platform. This opens up the possibility of having a centrally managed site-wide installation of an arbitrary set of VisualWorks versions, allowing users to simply run their images as executables without any user-specific setup required. The loader figures out which version of VisualWorks and which specific VM is needed to run the image being launched, using information provided in the INI file).

For installations using only final releases (not development build releases), a single entry line in the INI file for each VisualWorks version will suffice to serve any Unix-based platform for which a VM is available at the specified location.

MS-Windows Systems

The two Windows 4-byte loader files in `/preview/bin` have been renamed to `VisualWorksXL.exe` and `VisualWorksXL.ini`. The `.ini` file has been rewritten and commented to be clearer. The supported two-byte Windows loader, `VisualWorks.exe`, remains available in `/bin/win` as before.

Base Image for Packaging

/preview/packaging/base.im is a new image file to be used for deployment. This image does not include any of the standard VisualWorks programming tools loaded. The image is intended for use as a starting point into which you load deployment parcels. Then strip the image with the runtime packager, as usual.

BOSS 32 vs. BOSS 64

The current implementation of BOSS (boss32, version 7), does not accomodate 64-bit small integers and small doubles natively. Also, it does not support extremely large objects that are outside the implementation limits for 32 bits. Furthermore, since the implementation of `identityHash` is not equal in 32 and 64 bit images, identity based collections may require special handling when moving them across images of different bit size.

A preview implementation of boss64 (version 8) has been implemented for this purpose. This implementation is an addition to the existing BOSS parcel, and is called BOSS64.

The new BOSS implementation has been structured so that there is a new factory class that takes care of choosing the proper reader for either boss32 or boss64 without user intervention, and a similar factory arrangement that chooses either boss32 or boss64 as the output format depending on the image BOSS is running on.

More concretely, until now application code would have referred to `BinaryObjectStorage` to write BOSS streams in boss32 format:

```
BinaryObjectStorage onNew: aStream
```

Referencing the class `BinaryObjectStorage64` instead will result in BOSS streams in boss64 format:

```
BinaryObjectStorage64 onNew: aStream
```

Finally, referencing `AbstractBinaryObjectStorage` will choose either boss32 or boss64 depending on the image in which the code is running:

```
AbstractBinaryObjectStorage onNew: aStream
```

Moreover, referencing the abstract factory class for reading,

```
AbstractBinaryObjectStorage onOld: aStream
```

will automatically determine the format of the stream and choose the appropriate reader accordingly:

Execution environment	Selected reader
32-bit image, 32-bit BOSS stream	`BOSSReader`
64-bit image, 32-bit BOSS stream	`BOSSReader32`
64-bit BOSS stream	`BOSSReader64`

Existing code making reference to classes already present before these changes will not be affected, and they will still rely on existing boss32 behavior.

Also, although boss64 streams can be written by 32 bit images, 32 bit images should write BOSS streams in 32 bit format because 64 bit images can read these BOSS streams while doing all the necessary conversions.

64-bit Image Conversion

The `ImageWriter` parcel is still capable of converting arbitrary 32-bit images to 64-bit images. However, due to an unresolved race condition, occasionally it may create an image that brings up one or more error windows. These windows can safely be closed, and if the 64-bit image is saved again, they will not return.

However, they may be problematic in a headless image or an image that has been produced by the Runtime Packager. For such cases, re-saving or recreating the original 32-bit image, and then converting it again may avoid the race condition. Alternatively, converting the image to 64 bits before applying the Runtime Packager or making the image headless may also be helpful.

`ImageWriter` empties all instances of `HandleRegistry` or its subclasses. Since these classes have traditionally been used to register objects which must be discarded on startup, emptying them during the image write is safe. But if your code is using `HandleRegistry` or a subclass to hold objects which are intended to survive across snapshots, ImageWriter may disrupt your code. Running ImageWriter before initializing your registries may solve this problem. We would also like to know more about how you use HandleRegistry, in order to improve ImageWriter's ability to transform images without breaking them.

Tools

With the Trippy basic inspector now being much more robust, work was done on integrating this with the debugger. Nicknamed Diggy, it has not been finished yet, but may be loaded from **/preview/parcels**.

Note: Given the ongoing renovation of tools in VisualWorks 8.x, Diggy won't be updated for 8.2. If Cincom updates Diggy again it'll be to integrate it in to the product fully. It might go a different direction, though still with the intention of unifying the inspector tools, especially in the debugger.

At this writing, this causes the inspectors located in the bottom of the debugger (the receiver and context fields inspectors) to be basic trippy inspectors (not the entire diving/previewing inspector, just the basic tab). This makes operations between the two the same, and provides the icon feedback in the debugger. The stack list of the debugger also shows the receiver type icons.

Cairo

Going forward, Cairo will not be an officially supported framework. When the Graphics2 framework is complete, Cincom plans to obsolete Cairo.

That said, customers continue to use Cairo and Cincom has been updating it. In a sense, it is a preview to Graphics2, but only indirectly.

Some details on using Cairo are available in previous versions of the VisualWorks Release Notes.

Internationalization

MessageCatalogManager supports multiple locales simultaneously

The PerProcessCatalogs package, in preview (preview/parcels), extends the existing MessageCatalogManager facilities to support simultaneous use of multiple locales. This is mainly needed for application servers where different clients may require server side processing to use different catalogs depending on the client locale.

For languages with different variants in different regions, e.g. different english dialects #en_US, #en_GB, etc. the application may require some messages to be localized differently, e.g. 'color' vs. 'colour'. At the same time many other message are common among the regions. To support this efficiently, the Locale searches for the translations in the most specific catalogs first and then looks for less specific ones if that fails. For example, an #en_US Locale may subsequently look into catalogs for en_US, en and finally C.

Opentalk CORBA

This release includes an early preview of our OpentalkCORBA initiative. Though our ultimate goal is to replace DST, DST will remain a supported product until OpentalkCORBA matches all its relevant capabilities and we provide a reasonable migration path for current DST users. So, we would very much like to hear from our DST users, about the features and tools they would like us to carry over into OpentalkCORBA.

For example, we do not intend to port any of the presentation-semantic split framework, or any of the UIs that essentially depend upon it, unless there is strong user demand. Please contact Support, and ask them to forward your concerns and needs to the VW Protocol and Distribution Team.

This version of OpentalkCORBA combines the standard Opentalk broker architecture with DST's IDL marshaling infrastructure to provide IIOP support for Opentalk. OpentalkCORBA has its own clone of the IDL infrastructure residing in the Opentalk namespace so that changes made for Opentalk do not destabilize DST. The two frameworks are almost capable of running side by side in the same image. The standard base class extensions, however, like 'CORBAName' can only work for one framework, usually the one that was loaded last. Therefore, if you want to load both and be sure that DST is unaffected, make sure it is loaded after OpentalkCORBA, not before.

This version of OpentalkCORBA already offers a few improvements over DST. In particular, it supports the newer versions of IIOP, though there is no support for value types yet. A short list of interesting features and limitations follows:

- supports IIOP 1.0, 1.1, 1.2
- defaults to IIOP 1.2
- does not support value types
- does not support Bi-Directional IIOP
- doesn't support the `NEEDS_ADDRESSING_MODE` reply status
- system exceptions are currently raised as `Opentalk.SystemExceptions`
- user exceptions are currently raised as Error on the client side
- supports LocateRequest/LocateReply
- does not support CancelRequest
- does not support message fragmenting
- the general IOR infrastructure is fleshed out (`IOPTaggedProfiles`, `IOPTaggedComponents`, `IOPServiceContexts`) and adding new kinds of these components amounts to adding new subclasses and writing corresponding read/write/print methods
- the supported profiles are `IIOPProfile` and `IOPMultipleComponentProfile`, and anything else is treated as an `IOPUnknownProfile`
- the only supported service context is `CodeSet`, and anything else is treated as an `IOPUnknownContext`
- however it does not support the codeset negotiation algorithm yet; correct character encoders for both char and wchar types can be set manually on the `CDRStream` class
- the supported tagged components are `CodeSets`, `ORBType` and `AlternateAddress`, and anything else is treated as an `IOPUnknownComponent`

IIOP has the following impact on the standard Opentalk architecture and APIs:

- there is a new `IIOPTransport` and `CDRMarshaler` with corresponding configuration classes
- these transport and marshaler configurations must be included in the configuration of an IIOP broker in the usual way
- the new broker creation API consists of the following methods

- `#newCdrIIOPAt:`
- `#newCdrIIOPAt:minorVersion:`
- `#newCdrIIOPAtPort:`
- `#newCdrIIOPAtPort:minorVersion:`

- IIOP proxies are created using `Broker>>remoteObjectAt:oid:interfaceId:`
- there is an extended object reference class named `IIOPObjRef`
- the `LocateRequest` capabilities are accessible via

- `Broker>>locate: anIIOPObjRef`
- `RemoteObject>>_locate`

- LocateRequests are handled transparently on the server side.
- A location forward is achieved by exporting a remote object on the server side (see the example below)

Examples

Remote Stream Access

The following example illustrates basic messaging capability by accessing a stream remotely. The example takes advantage of the IDL definitions in the SmalltakTypes IDL module:

```
| broker stream proxy oid |
broker := Opentalk.BasicRequestBroker newCdrIiopAtPort: 4242.
broker start.
[ oid := 'stream' asByteArray.
 stream := 'Hello World' asByteArray readStream.
 broker objectAdaptor export: stream oid: oid.
 proxy := broker
  remoteObjectAt: (
   IPSocketAddress
    hostName: 'localhost'
    port: 4242)
  oid: oid
  interfaceId: 'IDL:SmalltalkTypes/Stream:1.0'.
 proxy next: 5] ensure: [broker stop]
```

Locate API

This example demonstrates the behavior of the “locate” API:

```
| broker |
broker := Opentalk.BasicRequestBroker newCdrIiopAtPort: 4242.
broker start.
[ | result stream oid proxy found |
 found := OrderedCollection new.
 "Try to locate a non-existent remote object"
 oid := 'stream' asByteArray.
 proxy := broker
  remoteObjectAt: (
   IPSocketAddress
    hostName: 'localhost'
    port: 4242)
  oid: oid
  interfaceId: 'IDL:SmalltalkTypes/Stream:1.0'.
 result := proxy _locate.
 found add: result.
 "Now try to locate an existing remote object"
 stream := 'Hello World' asByteArray readStream.
 broker objectAdaptor export: stream oid: oid.
 result := proxy _locate.
 found add: result.
 found
] ensure: [ broker stop ]
```

Transparent Request Forwarding

This example shows how to set up location forward on the server side and demonstrates that it is handled transparently by the client.

```
| broker |
broker := Opentalk.BasicRequestBroker newCdrIiopAtPort: 4242.
broker start.
[ | result stream proxy oid fproxy foid|
 oid := 'stream' asByteArray.
 stream := 'Hello World' asByteArray readStream.
 broker objectAdaptor export: stream oid: oid.
 proxy := broker
  remoteObjectAt: (
   IPSocketAddress
    hostName: 'localhost'
    port: 4242)
  oid: oid
  interfaceId: 'IDL:SmalltalkTypes/Stream:1.0'.
 foid := 'forwarder' asByteArray.
 broker objectAdaptor export: proxy oid: foid.
 fproxy := broker
  remoteObjectAt: (
   IPSocketAddress
    hostName: 'localhost'
    port: 4242)
  oid: foid
  interfaceId: 'IDL:SmalltalkTypes/Stream:1.0'.
 fproxy next: 5.
] ensure: [ broker stop ]
```

Listing contents of a Java Naming Service

This example provides the code for listing the contents of a running Java JDK 1.4 naming service. It presumes that you have Opentalk-COS-Naming loaded. To run the Java naming service, just invoke 'orbd -ORBInitialPort 1050' on a machine with JDK 1.4 installed.

Note that this example also exercises the LOCATION_FORWARD reply status, the broker transparently forwards the request to the true address of the Java naming service received in response to the pseudo reference 'NameService'.

```
| broker context list iterator |
broker := Opentalk.BasicRequestBroker newCdrIiopAtPort: 4242.
broker passErrors; start.
[ context := broker
  remoteObjectAt: (
   IPSocketAddress
    hostName: 'localhost'
    port: 1050)
  oid: 'NameService' asByteArray
  interfaceId: 'IDL:CosNaming/NamingContextExt:1.0'.
 list := nil asCORBAParameter.
 iterator := nil asCORBAParameter.
 context
  listContext: 10
  bindingList: list
  bindingIterator: iterator.
 list value
] ensure: [ broker stop ]
```

List Initial DST Services

This is how you can list initial services of a running DST ORB. Note that we're explicitly setting IIOP version to 1.0.

```
| broker dst |
broker := Opentalk.BasicRequestBroker
  newCdrIiopAtPort: 4242
  minorVersion: 0.
broker start.
[ dst := broker
  remoteObjectAt: (
   IPSocketAddress
    hostName: 'localhost'
    port: 3460)
  oid: #[0 0 0 0 0 1 0 0 2 0 0 0 0 0 0 0]
  interfaceId: 'IDL:CORBA/ORB:1.0'.
 dst listInitialServices
] ensure: [ broker stop ]
```

International Domain Names in Applications (IDNA)

RFC 3490 “defines internationalized domain names (IDNs) and a mechanism called Internationalizing Domain Names in Applications (IDNA) which provide a standard method for domain names to use characters outside the ASCII repertoire. IDNs use characters drawn from a large repertoire (Unicode), but IDNA allows the non-ASCII characters to be represented using only the ASCII characters already allowed in so-called host names today. This backward-compatible representation is required in existing protocols like DNS, so that IDNs can be introduced with no changes to the existing infrastructure. IDNA is only meant for processing domain names, not free text” (from the RFC 3490 Abstract).

Limitations

The current implementation in VisualWorks:

- doesn't do NAMEPREP preprocessing of strings (currently we just convert all labels to lowercase)
- doesn't properly implement all punycode failure modes
- needs exceptions instead of Errors
- needs I18N of messages

Usage

You can convert an IDN using the `IDNAEncoder` as follows:

```
IDNAEncoder new encode: 'www.cincom.com'
  "result: www.cincom.com"
```

or

```
IDNAEncoder new encode: 'www.cìncòm.com'
 "result: www.xn--cncm-qpa2b.com"
```

and decode with

```
IDNAEncoder new decode: 'www.xn--cncm-qpa2b.com'
 "result: www.cìncòm.com"
```

This package also overrides the low level DNS access facilities to encode/decode the hostnames when necessary. Here's an example invocation including a Japanese web site.

```
host := (String with: 16r6c5f asCharacter with: 16r6238 asCharacter), '.jp'.
address := IPSocketAddress hostAddressByName: host.
 "result: [65 99 223 191]"
```

The host name that is actually sent out to the DNS call is:

```
IDNAEncoder new encode: host
 "result: xn--0ouw9t.jp"
```

A reverse lookup should also work, however at release time we were unable to find an IP address that would successfully resolve to an IDN. This functionality remains untested, and even our example gives only the numeric result:

```
IPSocketAddress hostNameByAddress: address
 "result: 65.99.223.191"
```

MatriX

MatriX

MatriX (not to be confused with Polycephaly) provides simple mechanisms for spawning multiple running copies of a single image and using those to perform various tasks in parallel. This is primarily useful when attempting to utilize hosts with multi-core CPUs. The images are spawned headless and are connected with the main controlling image through their standard I/O streams, which are wrapped with BOSS so that arbitrary objects can be sent through.

There are two versions of MatriX in preview at this time. MatriX (polycephaly2.pcl) is a direct descendant of Polycephaly 1 (polycephal.pcl), however there are some key differences between the two frameworks.

Under the hood, Polycephaly 1 used BOSS to marshal objects, while MatriX uses the Xtreams ObjectMarshaler. These two marshalers may handle object edge cases differently.

Polycephaly 1 used pipes to communicate with local images, while Polycephaly 2 uses sockets. On Unix platforms, these are domain sockets, which should be secure — but on Windows they may be TCP sockets, which may NOT be secure.

For more information and examples, refer to the package comments.

WWW

HTTP2

In VisualWorks 8.2, the package `SiouX-Http2` provides an implementation of an HTTP2 server based on RFC 7540.

The implementation is functional, though it is currently missing support for Stream Priority and Dependency. As it stands, there is only OpenSsl support for the TLS1.2 Cipher Suites for HTTP2.

To create SiouX server with HTTP2 support over a clear connection:

```
server := Server id: 'Http2ServerTest'.
server addResponder:
    (responder := Hello new path: '/hello'; yourself).
listener := server listenOn: 8000 for: HttpConnection.
listener protocolVersions: (Array with: HTTPv20 new).
server start.
```

To create a SiouX server with HTTP2 support over a secure connection:

```
Crypto.LibCryptoEVP bePreferredLibrary.
server := Server id: 'TestH2SecureServer'.
server addResponder: (H2Hello path: '/').
listener := server listenOn: 8002 for: SiouX.HttpsConnection.
certificates := Xtreams.TLSCertificateResource current serverCertificates.
serverContext := TLSContext newServerWithDefaults
        certificates: certificates;
        yourself.
serverContext suites: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256.
alpn := Xtreams.TLSAppLayerProtocolNegotiation defaults.
serverContext extensions: (Array with: alpn).
listener protocolVersions:
    (Array with:
     (Protocols.HTTPv20 new
      settings: Protocols.SettingsFrame default; yourself)).
listener tlsContext: serverContext.
server start.
```

Chapter

4

Deprecated Features

Topics

- Database
- GUI

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.

Database

String parameter to ConnectionProfile has been deprecated

Class ConnectionProfile expects that the parameter provided to #driverClassName: will indeed be a possible class name, i.e., a Symbol. Providing a String still works, but raises a deprecation warning. (Users are enouraged to migrate any callers that provide String values.)

GUI

UILook support is obsolete

In this release, all UILook and supporting classes for MS-Windows, OS2, and OS X are obsolete. In place of these, the UISkinning framework now provides native look support on Apple's OS X or via the MS-Windows `UxTheme` DLL to provide a high-fidelity, native look for those platforms. A Default skin available on any platform provides a platform-agnostic look for X11.

Chapter

5

Legacy

Topics

There are some frameworks in VisualWorks that are obsolete, antiquated, or have been replaced by newer technology, and have been designated as “Legacy”.

We keep some of these frameworks available in the product, as some customers may want to use them longer, need extra time to port to a newer technology, or may wish to continue with and maintain the framework for themselves.

Legacy frameworks have minimal to no support and functionality will not be enhanced.

Over time, these frameworks may move to an obsolete designation, or be removed from the distribution. For feedback concerning your use of any legacy components, please email: Arden Thomas <athomas@cincom.com>and Suzanne Fortman <sfortman@cincom.com>.

Legacy Frameworks

The following list includes the name of the framework, and a brief note explaining the rationale for its legacy status.

VisualWorks/Foundation

Foundation refers to the core that is used by both VisualWorks and ObjectStudio.

Distributed Smalltalk/CORBA

MQSeries

VisualWave

Supplanted by SiouX, AppeX, and Seaside.

ObjectLens

Supplanted by Glorp.

“Old” Internationalization Locale for Japanese

Use the new Internationalization framework for upgrades and new application development.

ObjectStudio

Model Editor

SQL Editor

Report Writer

Database

Sqlparser

In the `Proxy-Database` package.

Scrollable cursor classes

Delete and update methods.