Cincom®

Cincom Smalltalk™

VisualWorks®

Release Notes

VisualWorks 8.3

P46-0106-26

SIMPLIFICATION THROUGH INNOVATION

Notice

Part Number: P46-0106-26

Software Release: 8.3

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 8.3

Topics

These release notes outline the changes made in VisualWorks 8.3. Both Commercial and Non-Commercial (PUL) 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

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 option, check that the bundle's own prereqs are what the parcel will need, since the prereqs of its sub-pundles 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.

Chapter

2

New and Enhanced Features

Topics

This chapter describes the new functionality and major changes in this release of VisualWorks.

Base Image

Clarification on End of Support for Short Compiled Code

As noted in the release notes to previous versions of VisualWorks, the Smalltalk compiler was modified in version 7.7, such that it would no longer generate "short compiled code". With VisualWorks 8.0, support for short compiled code was removed from the virtual machine but a provision was made to automatically convert any short compiled code upon load. With release 8.2, this auto-convert-on-load support was definitively removed, completing the transition of the compiler, virtual machine, and Store.

However, the release notes to VisualWorks 8.2 stated erroneously that pre-7.7 code which had been published in parcel form would still be automatically converted on load. In fact, this is no longer the case. To clarify, then, in versions 8.2 and later of VisualWorks, attempting to load parcels that were published from a version 7.6 or earlier release may report the following errors in Store: **Invalid parcel file format** and **Load aborted.**

If you need to load a pre-7.7 parcel into an 8.2 or later image, you can first load it into any version of VisualWorks between 7.7 and 8.1.1, republish the parcel, and then load the republished code into an 8.2 or later image. For details on short compiled code and the rationale for removing it, please refer to the *VisualWorks Release Notes* for versions 7.7 and 8.2.

Enhanced Functionality for Parcels

The support for publishing and loading overrides in parcels has been enhanced.

Multiple parcels with extensions to an ExternalInterface class can now be published and loaded successfully. Previously, multiple extensions to ExternalInterfaces could not be loaded together unless they had been developed completely separately.

ImageWriter Enhancements

With VisualWorks 8.3, ImageWriter is more reliable running in 64-bit images, and should be able to convert 64-bit images to 32-bit images in addition to the reverse.

GUI

Support for UI Painter Custom Colors

When UI Skinning was first introduced, custom colors from the UI Painter's color tab were not supported. VisualWorks 8.3 brings back support for custom colors in the UI Painter, and color selections that already exist in window specs will now work.

Because Skinning's new look-and-feels completely replace the look-and-feels of VisualWorks 7 and earlier, support for custom colors is not precisely the same, especially for native look-and-feels. For the most complete coverage of custom colors, select the **Emulated Windows** skin, or one of the new Classic Skins (for these, load the `UISkinning-Skins-Classic` parcel), on the **Look and Feel** page of the Settings Tool. The default skins are also a good choice if you need custom colors.

The UI Painter also now permits the coloring of widgets again, and its functionality is enhanced with **Select** and **Clear** buttons below the color tiles.

Interim Support to Remove Size Restrictions on Bordered/Unbordered Action Buttons

With the introduction of UI Skinning, buttons are restricted to standard platform heights since on Windows and OS X platforms bordered buttons are rendered using the platform-standard bounding graphics. VisualWorks 8.3 provides a new class, `LegacyActionButtonSpec`, which can be used as a workaround to define `ActionButton`s whose bounds honor the layout expectations of prior releases as coded in the window spec.

To use this workaround, update the button definition in existing window specs to replace `ActionButtonSpec` with `LegacyActionButtonSpec`. This can be done either using a rewrite rule or by checking the

Use legacy button layout option on the **Details** page for the button in the UI Painter. Changing the spec class will configure the button to use either the LegacyPushButtonArtist (for bordered buttons) or LegacyUndecoratedPushButtonArtist (for unbordered buttons). These artists do not restrict the button to a platform standard size, but draw it using the bounds defined by its layout in the window spec. The LegacyActionButtonSpec class is temporary, but its use in window specs will be compatible with future releases that incorporate the new layout framework. The corresponding artist classes are strictly temporary to this workaround, since we expect artists and skins to continue to evolve as we move toward new, variant and custom widgets.

Migration from 8.x to 8.3

Prior to VisualWorks 8.3, Text2 widgets erroneously treated autoAccept as continuousAccept. This has been fixed, but any window spec using a DocumentViewSpec, DocumentEditorSpec, InputEditorSpec or SourceCodeEditorSpec may need to be updated. If autoAccept: false is specified, this line of code should be removed so that the default value of true will be used.

Database

ODBC3Connection becomes the default ODBCConnection

In VisualWorks 8.3, ODBC 3 behavior becomes the default associated with ODBCConnection.

If you first connect using an ODBC2Connection, the environment for the whole image will be set to behave like ODBC 2 and this will still be the case even if you later connect using an ODBCConnection, i.e. that new connection will also have ODBC 2 behavior. If you want to change to ODBC 3 behavior, you must first disconnect all ODBC2Connection instances in the image; only then will the environment handle be freed. After that, if you connect an ODBCConnection, a new environment handle will be created and set to behave like ODBC 3. If you want to change from ODBC 3 to ODBC 2 behavior, the same rule applies.

Moreover, when executing batched SQL statements, ODBC 2 and ODBC 3 work differently: ODBC 2 only returns an answer for a SQL statement which results in a result set, while ODBC 3 returns an answer for every SQL statement.

To illustrate the behavior of ODBC 2, consider the following example:

```
"Connect to the server."
conn := ODBC2Connection new.
conn username: 'username';
  password: 'password';
  environment: 'env'.
conn connect.

"Drop the test table if it already exists."
sess := conn getSession.
sess prepare: 'DROP TABLE TestTable'.
sess execute.
ans := sess answer.

"Create a test table."
sess prepare: 'CREATE TABLE TestTable(
            cid int,
            cname varchar(100))'.
sess execute.
ans := sess answer.

"Run a batch of SQL statements."
sess := conn getSession.
sess prepare: 'INSERT INTO TestTable VALUES (?, ?)
select * from TestTable
select cid from TestTable
'.
sess bindInput: (Array with: 1 with: 'test1').
sess execute.

"Calling #answer, to get the first result set."
ans := sess answer.
rows := ans upToEnd.

"Get the second result set."
ans := sess answer.
rows := ans upToEnd.
```

```
"No more answers (ie., ans == #noMoreAnswers)."
ans := sess answer.
```

The following example demonstrates the ODBC 3 behavior:

```
"Connect to the server."
conn := ODBCConnection new.
conn username: 'username';
  password: 'password';
  environment: 'env'.
conn connect.

"Drop the test table if existed."
sess := conn getSession.
sess prepare: 'DROP TABLE TestTable'.
sess execute.
ans := sess answer.

"Create a test table."
sess prepare: 'CREATE TABLE TestTable(
            cid int,
            cname varchar(100))'.
sess execute.
ans := sess answer.

"Run a batch of SQL statements."
sess := conn getSession.
sess prepare: 'INSERT INTO TestTable VALUES (?, ?)
select * from TestTable
select cid from TestTable
'.
sess bindInput: (Array with: 1 with: 'test1').
sess execute.

"No answer set for the first SQL (ans == #noAnswerStream)."
ans := sess answer.

"Get the first result set."
ans := sess answer.
rows := ans upToEnd.

"Get the second result set."
ans := sess answer.
```

```
rows := ans upToEnd.

"No more answers (ans == #noMoreAnswers)."
ans := sess answer.
```

Glorp

Glorp Performance Improvements

Relative Attributes

High-performance applications that frequently update numeric fields can now do so reliably in a single update, without requiring optimistic locking. Sending `beRelative` to a `DatabaseField` causes it to update the target column by changing that value the same amount the instance has changed. For example, if the instance value has increased by `10`:

```
UPDATE qty SET qty = qty + 10 WHERE ... RETURNING qty
```

This relative update preserves the target column’s integrity, and updates the in-image instance with the resulting value, such that the field stays in synch with the target column. For more details, see section "Using Relative Fields" in the *Glorp Developer's Guide*.

Recursion

In VisualWorks 8.3, Glorp can combine ordinary queries to define a recursive query that will run on databases that support recursion (and fail if they do not). This feature was previewed in 8.2. It has been developed further and integrated in the Glorp framework. For details, see "Recursive Queries" in the *Glorp Developer's Guide*.

Glorp Functional Impovements

In VisualWorks 8.3, Glorp includes a number of code improvements and fixes, including enhancements to support for conditional mappings. To summarize, conditional mappings can have buildable as well as simple objects in their various cases. Your application can call the new method `setUpBaseFromSession:` on a query and then send `retrieve:`, `AND:` or `OR:` with blocks that use aggregation calls, e.g. `isEmpty`, `sqlCount` (previously such calls would have failed). A bug that made it difficult to use an attribute that was declared writable but not readable, or readable but not writable, has been fixed. When compound statements also fetch fields from tables mapped to objects related to the base query object, they can order or group by those fields, and alias same-name fields. The new `alias:` function helps developers to alias expressions in Glorp blocks (note: SQLServer requires that retrieved relational expressions be aliased). In previous releases, if two subqueries in a `CompoundStatement` returned two different constants, an over-eager optimization returned the same value for both. This has been fixed. Schemas can now map inheritance horizontally (i.e. separate tables for subclasses) while using `VariableJoins`. Query where clauses can accept code like:

```
[:each || subquery |
    subquery := Query read..... .
    ... other conditions ... AND:
    [(each getMappings: #(name timestamp)) in: subquery]]
```

i.e. you can define your own composite to compare with `in:`, `=` and `<>`. (Previously, an object with a composite primary key could compare in this way but user-defined mappings were not supported.)

The `GlorpTest` parcel has usage examples for all of these enhancements.

Internationalization

Enhancements to Internationalization Support

With the release of VisualWorks 8.3, internationalization support has been updated to the latest released version (30.0.3) of the Common Locale Data Repository (CLDR) from the Unicode Consortium.

The CLDR project is the most authoritative source of internationalization information in software development today, and is used by nearly all developers either directly or indirectly, in almost all computer languages now popular. The ICU (International

Components for Unicode) package, also based on CLDR, is the source for most internationalization libraries for C, C++, and Java.

With each release, there are changes in the data content of the CLDR. Locales are added and removed. Various other data items are updated or replaced.

In the current release of VisualWorks (8.3), incorporating CLDR 30.0.3, the most notable changes from our previous release are:

- In print policy formats in classes such as `TimestampPrintPolicy` and `TimestampReader`, seconds are no longer present in the time portion of textual representations created using the `#short` format. This makes this format a poor choice for those planning on a round trip with their data: for example, from `Timestamp` to a textual representation and eventually back to `Timestamp`. Seconds remain in the formats for ISO-8601 representations in all `Locale`s, and in formats in the `C Locale`, which have been historically accessed for several functions internally within VisualWorks.
- In print policy formats in classes such as `TimestampPrintPolicy` and `TimestampReader`, a word is now added between the date and time portions of textual representations created using the `#full` and `#long` formats. This word or phrase is language-appropriate for the `Locale`, so in English a timestamp textual representation created with these formats might read:

  'February 14, 2017 at 2:30:00 PM'

  In German, by contrast, the preposition is "um" rather than "at", and of course the month name would also be in German. The method `Timestamp class>>readFromDateAndTime:` (which reads from a textual representation in a stream) has been enhanced to step over the preposition, if present, regardless of the national language used in the textual representation.

It's also appropriate to remind folks at this time that there has been support for the creation and use of a `#custom` format since the previous release, and this is documented in the *VisualWorks Internationalization Guide*. This is the first suggestion for developers who need a persistent format that includes seconds (or fractional seconds), or want other customizations in their textual representation creation.

Also note that there is hinted reading for such formats, with the #custom format enjoying the bias of being checked first if it exists. Of course, use of the C Locale is another option, but as the method to activate this support is to set the Locale of your operating system to the C Locale, most users won't opt for this.

Another option is to use a Locale in a separate Smalltalk process, now that per-process Locales are standard in the image, and this is also documented in the *Internationalization Guide*.

Since every Locale has a personalized print policy, and every print policy gives access to an appropriate reader, use of information from a Locale other than the system locale, such as the C Locale, may also be accessed with code such as the following:

```
| aStream sourceTimestamp retrievedTimestamp |
aStream := ReadWriteStream on: String new.
sourceTimestamp := Timestamp now.
(Locale named: #C) timePolicy
 print: sourceTimestamp on: aStream policyNamed: #long.
retrievedTimestamp := (Locale named: #C) timePolicy reader
 readLocaleTimestampFrom: (aStream reset).
```

Because the C Locale includes seconds and fractional seconds in its #long format, sourceTimestamp and retrievedTimestamp contain the same values, which extend to the fractional seconds. This example has the benefit that the Locale in use doesn't need to be changed, but the Locale must be used explicitly in this fashion for both reading and writing of textual representations.

Forthcoming support for ISO-8601 will provide for use as a storage format for round-trip data paths, as well as other needs for unambiguous transfer of date and time information. There will be a notice when that support is available.

Net Clients

Support for the Subject Alternative Name Extension

The Subject Alternative Name Extension allows identities to be bound to the subject of a certificate. These identities may be included in addition to or in place of the identity in the subject field of the certificate.

The VisualWorks implementation is based on RFC 5280 4.2.1.6.

A secure HTTP client can verify the subject alternative names when a TLS connection is established. To retrieve the alternative names, a client sends subjectAltNames to the certificate object in the tlsSubjectVerifier block. The method returns a Collection of Associations, where the key is one of GeneralName (see the specification) choices and the value is a correspoding object.

For example:

```
client := HttpClient new.
client tlsSubjectVerifier: [:certificate |
 certificate subjectAltNames do: [:association |
  "Add custom verification here"
  Transcript cr; show: association printString].
  true].
 [client get: 'https://google.com' ]
  on: Xtreams.TLSUnknownCA
  do: [:ex | ex resume].
```

When this example code is evaluated, the Transcript displays the Association objects, which contain dNSName name strings, e.g.:

```
#dNSName -> '*.google.com'
#dNSName -> '*.android.com'
#dNSName -> '*.appengine.google.com'
#dNSName -> '*.cloud.google.com'
....
```

Class Mailbox Supports Secure Connections

In VisualWorks 8.3, class Mailbox has been enhanced to support secure connections. To use a secure connection, create a HostSpec with the option #beSecure, e.g.:

```
host := HostSpec protocol: #IMAP host: 'mysecurehost' user: 'userid'.
host beSecure.
mailbox := IMAPMailbox user: host user server: host.
[mailbox messageCount]
```

```
on: Xtreams.TLSCertificateWarning
do: [:ex | ex proceed].
```

To avoid handling TLS certificate warnings for every mailbox command, the mailbox can be set with a `certificateWarningBlock`. This is a one-argument block to validate and take action for any TLS certificate warnings. The block argument is `TLSCertificateWarning`. To proceed with the connection, the block must return `true`. To illustrate, a secure mailbox can be created as follows:

```
mailbox := IMAPMailbox user: host user server: host.
mailbox certificateWarningBlock: [:warning | "add warning processing" true].
```

Alternately, the block can be specified via a global setting:

```
Mailbox class>>certificateWarningBlock: aBlock
```

By default, there is no TLS warning handling, as the default certificate warning block returns `false`.

Store

Store Enhancements

Schema Creation

The creation of schemas on Oracle has been improved such that all of the Store indexes will be created in the same tablespaces as their tables. Previously, indexes were created in the `USERS` tablespace.

In addition, the use of the `BLOB` column type has been eliminated on Oracle in favor of `LONG RAW`. Although `LONG RAW` has been deprecated by Oracle, it is supported on Oracle 12 and offers signifcantly better performance than `BLOB`. In the near future, we intend to replace `LONG RAW` with Oracle 12's larger `VARCHAR` capabilities.

Unloading

When unloading a package or bundle from Store, the Tools will now warn you before removing class extensions made obsolete by the removal of a class defined in the pundle being unloaded.

Thus, it is now possible to unload StoreBase from an image.

Option to Use Recursive SQL

The Store settings page now includes an additional option (which is turned on by default):

Read long methods (>32k source) faster

When this option is enabled, Store reads all of the source of long methods in a single round trip (if the database supports recursive SQL). This greatly speeds the reading of methods with more than 32k of source, which are generally quite rare but might be common in a given application, at the possible cost of slightly complicating the reading of other methods.

Security

Support for Extended Master Secret Extension (RFC 7627)

VisualWorks 8.3 includes support for Extended Master Secret Extension (RFC 7627). The use of EMSE improves TLS security by eliminating the possibility of a Man-In-The-Middle attack where a malicious web server could synchronize session parameters between the victim client and server, allowing the attacking web server to spoof a legitimate web site.

Class TLSServerContext now includes TLSExendedMasterSecret in the defaultExtensionsValue class method, making the use of EMSE automatically if the client sends the extension in the initial ClientHello message. No special action is required by your applications unless you wish to disable EMSE. In that case, either the TLSServerContext defaultExtensions class instance variable or the extensions instance variable in the TLSServerContext associated with a particular HttpsListener must be set by the user application, depending on the application requirements and specific environment.

References

Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension (RFC 7627).

A plain language explanation of the attack scenario: TLS Extended Master Secret Extension: Fixing a Hole in TLS.

Support for Server Name Indication (SNI)

In VisualWorks 8.3, we have added support for the "Server Name Indication" (SNI) TLS Extension, as originally defined in RFC 3546 (§ 3.1), with an update in RFC 6066 (§ 3). SNI is a mechanism to mitigate Man-In-The-Middle (MITM) attack attempts.

SNI improves TLS security by specifying the name of the server that is allowed to participate in a TLS handshake. For servers, this provides an opportunity to enforce an exact match between a requested server name and a name the server supports (for example, in a virtual host scenario). For clients, it is an additional mechanism to verify the authenticity of the server with which they are establishing a connection.

The VisualWorks 8.3 version of the `TLS` parcel includes some important additions to the protocol implementation. A new class named `TLSServerName` implements the extension itself. Class `TLSContext` has been split into two separate subclasses, `TLSClientContext` and `TLSServerContext`, to allow for the difference in behavior related to the use of SNI.

When upgrading to VisualWorks 8.3, developers of applications that directly manipulate the `tlsContext` in their client or server configurations are advised to review class `TLSContext` and the APIs of its subclasses to take advantage of SNI, and to avoid backward-compatibility issues.

On the client side, you can add SNI support by sending the message `sniHostNames:` to an instance of `TLSClientContext`. The argument should be a `Collection` (typically an `Array`) of server names that you expect the server to handle.

On the server side, you can add SNI support by sending the message `acceptedServerNames:` to an instance of `TLSServerContext`. The argument should be a `Collection` of server names that the server will accept, refusing TLS connections from clients that do not provide SNI or have no matching server name in the SNI extension of the initial `ClientHello` message during the TLS handshake. If `acceptedServerNames:`

has not been sent to a `TLSServerContext`, the server will ignore the client SNI, and will accept TLS connections from any client.

References

Transport Layer Security (TLS) Extensions (RFC 3546 § 3.1)

Transport Layer Security (TLS) Extensions: Extension Definitions (RFC 6066 § 3)

DotNETConnect

DotNETConnect updated for 8.3

DotNETConnect now uses version 4 of the .NET framework. If you only connect to assemblies that use version 4.x of the framework, you no longer need to set the `useLegacyV2RuntimeActivationPolicy` flag. If you use assemblies for framework versions 2.x or 3.x, you now need to set that flag.

Virtual Machine

Enhanced support for Retina Displays on OS X

Support for Retina displays on OS X is now supported rather than in `/preview`.

Support for 64-bit VMs on OS X and Linux PPC

64-bit virtual machines for OS X and Linux PPC — which were in `/preview` in VisualWorks 8.2 — are now a fully-supported part of the product.

Improved loading of 64-bit images on MS-Windows

We have improved the ability to load 64-bit images on Windows without running afoul of Microsoft's memory randomization mechanism, which was a problem in VisualWorks 8.2.

WWW

Support for HTTP/2

In VisualWorks 8.3, the SiouX framework includes support for the HTTP/2 protocol, as specified by RFC 7540. This functionality, formerly in **/preview**, has been fully integrated with the core SiouX framework. The components of the framework have been augmented with the Protocols-Http2 and SiouX-Http2 parcels. The basic architecture of the framework remains unchanged.

For details on the components, APIs, and examples of how to work with HTTP/2, please refer to the *Web Server Developer's Guide*.

Support for Server Name Verification

Class HttpListener has been enhanced with a new option acceptedServerNames. According to RFC 7230 § 5.4, a server must respond with a 400 (Bad Request) status code to any HTTP/1.1 request message that lacks a Host header field, and to any request message that contains more than one Host header field or a Host header field with an invalid field-value.

An instance of HttpListener can set a list of server names that are acceptable for incoming connection requests. The specified names are used to validate the Host header field. By default the collection is empty and the Host header field is not validated. To set acceptable server names:

```
server := Server id: 'TestServer'.
server addResponder: Hello new.
listener := server listenOn: 8000 for: HttpConnection.
listener acceptedServerNames: #('localhost').
```

If an HTTP/1.1 request arrives with a Host header other than than 'localhost' the server returns a 400 Bad Request. If the request uses the HTTP/2 protocol, the :authority pseudo-header is checked against acceptedServerNames.

In the case of a secure connection, two-step validation is performed.

- Acceptable server names are set in the server tlsContext as a TLSServerName extension, and validated during the TLS handshake.
- Next, the HTTP layer validates the Host header for HTTP/1.1, or the :authority pseudo-header for HTTP/2.

For example:

```
server := Server id: 'TestServer'.
server addResponder: Hello new.
listener := server listenOn: 8000 for: HttpsConnection.
listener acceptedServerNames: #('localhost').
```

Help

Help2

VisualWorks 8.3 includes an all-new Help System built using the Text2 framework, for rendering content from industry-standard DITA-XML files. Following the DITA model, the Help content is organized as a tree of modular topics. Help2 has been integrated with the IDE tools (as before, via the **Help** menu) and fully supplants the old Help System based on the Arbor framework and proprietary-format XML help files.

In addition to rich content display, the new Help System includes an integrated high-speed full text search facility, a collapsible structured view on the content, breadcrumb display of the topic tree, automatic code formatting/highlighting for examples, basic support for tables in the DITA content, display of images, ability to execute inlined Smalltalk code via hyperlinks, ability to open code browsers from classes or components tagged with <apiname>, and the ability to open URLs like topic://vw.tools.inspector.

The source files are expected to be DITA format (i.e., **`*.ditamap`** and **`*.xml`**), and the Help System supports a basic subset of the version 1.3 specification. We make use of one noteworthy DITA convention, which is that the @otherprops attribute may be used to annotate several DITA elements. For example, with <codeblock> elements @otherprops can specify a source type such as "smalltalk method" or "smalltalk expression" and the Help Browser uses this as a cue to selectively format and highlight code examples. The `@otherprops` attribute can also be (optionally) added to `<apiname>` elements, as a notation for the species of browser to invoke when the hyperlink is activated; values include: `binding`, `pundle`, `implementors`, or `senders`.

In addition to the new Help Browser, the Help content has been reorganized and updated. Ivan Tomek's "Introduction to VisualWorks Smalltalk" has been updated and integrated in the Help System (formerly, it appeared as a Workspace in VisualWorks PUL). If you wish to use the new Help System as the basis for your applications' online help facility, feel free to contact us for additional details.

Documentation

Documentation Updates

This section provides a summary of the main documentation changes.

Application Developer's Guide

The chapter on Debugging has been rewritten to reflect significant changes in the IDE tools. Documented special requirements for deployment on MS-Windows platforms. Enhanced the discussion of the `Core.*` name space. New topic for troubleshooting parcel loading issues. Updated the discussion of the Change Set Manager. Enhanced discussion of pragmas. Enhanced the discussion of VM backward compatibility. Minor updates and corrections. Converted to use the new documentation template.

Basic Libraries Guide

Added a new, explanatory section "Line-end conversion in VisualWorks" to the existing topic "Line-end Conventions". Added a new topic "Customizing Entity Resolution" to the chapter on XML. Minor editorial corrections. Converted to use the new documentation template.

COM Connect Guide

No changes.

Database Application Developer’s Guide

Documented Restrictions on Array Binding with Batched Queries. Minor updates and corrections. Converted to use the new documentation template.

DLL and C Connect Guide

Minor editorial corrections. Converted to use the new documentation template.

DotNETConnect User’s Guide

Framework compatibility statement updated. Minor editorial corrections. Converted to use the new documentation template.

DST Application Developer’s Guide

No changes.

Glorp Guide

New documentation on support for recursive queries and relative fields. Minor editorial corrections. Converted to use the new documentation template.

GUI Developer’s Guide

No changes.

Internationalization Guide

Minor updates and corrections. Converted to use the new documentation template.

Internet Client Developer’s Guide

Minor updates and corrections. Converted to use the new documentation template.

MQ-Interface Guide

No changes.

Opentalk Communication Layer Developer's Guide

No changes.

Security Guide

Minor updates and corrections. Converted to use the new documentation template.

Source Code Management Guide

No changes.

Tool Guide

Updated the discussion of the Change Set Manager. Amended the chapter on the rewrite editor. Reorganized chapters. Numerous minor updates and corrections. Converted to use the new documentation template.

Walk Through

No changes.

Web Application Developer’s Guide

Minor updates and corrections. Converted to use the new documentation template.

Web Server Developer’s Guide

Support for HTTP/2 documented. Minor updates and corrections. Converted to use the new documentation template.

Web Service Developer’s Guide

Enhance discussion of how to create binding classes. Minor updates and corrections. Converted to use the new documentation template.

Chapter

3

Preview Components

Topics

Some features are included in the VisualWorks distribution on a "beta test," or even pre-beta test, basis, allowing us to provide you early access to forthcoming features.

This "preview" functionality is located in the `/preview` directory of the distribution, and described in the following sections of this chapter. You might also browse the contents of `/preview` for last-minute inclusions.

Base Image

Improved Graphics on Hi-DPI Displays

VisualWorks 8.3 includes a preview parcel `HiDef-Windows` which enables improved graphics on hi-dpi displays on Windows, comparable to retina support on OS X, but it is known to be incomplete. It does not require a preview VM.

GUI

Upgrading an Existing Application using Classic Skins

VisualWorks 8.3 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 contemporary look similar to that of Windows 10. These skins are usable either on their native platform or in an emulated form on a different platform. They have been created to honor the text attributes and font expectations of applications developed with pre-skinning versions of VisualWorks.

This is still preview code, but several improvements have been made for this release:

- The full set of standard text styles, System, Small, Large, Fixed, Named fonts, can be used in widget or UI definitions.
- Changing the skin will update the text style for widgets in all open UIs.
- If you remove the `UISkinning-Skins-Classic` package from an image which is currently using one of the classic skins, the image will revert to using the platform native skin.

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 VisualWorks tools do not yet make use of the new layout framework, so there are still a number of anomalies related to layout between these skins and their 7.x counterparts.
- Custom text style definitions other than the standard set cannot yet dynamically adjust when changing skins or platforms.

We welcome feedback about whether this compatibility package looks promising to customers upgrading from 7.x versions of VisualWorks. Our current plans are to incorporate the new layout framework as soon as possible, and improve the ability of text styles (both fonts and text attributes) to adjust to changing skins or platforms. We also have plans to add the ability to specify a skin at the UI and possibly the widget level.

Virtual Machine

Preview 64-bit Virtual Machines with Enhanced support for Perm Space

All 64-bit platforms have a preview VM in addition to the production VM. The preview VMs (located in `$VISUALWORKS/preview/bin`) support a perm-save option which will populate Perm Space. A VisualWorks image in which a significant portion of the objects are in Perm Space can do garbage collection much more quickly, although the garbage collection may be less effective if there are many objects in Perm Space that could have been collected if they were not being skipped.

Any image saved by a 64-bit preview PermSpace VM can't be loaded by a production VM. The AIX 64 VM, being available only as a preview, is only available as PermSpace-capable, so if you save images on AIX 64, you can't run them on other 64-bit platforms unless you use the preview VMs for those other platforms.

Preview Virtual Machines for Linux ARM / Raspberry Pi and compatibles

VisualWorks 8.3 includes a preview VM for Linux ARM / Raspberry Pi and compatibles. It is present as a separate option in the Installer, and the VM files can be found in: `$VISUALWORKS/preview/bin/linuxARM`

Preview Virtual Machine for 64-bit AIX

VisualWorks 8.3 includes a preview VM for 64-bit AIX. It is present as a separate option in the Installer, and the VM files can be found in: `$VISUALWORKS/preview/bin/aix64`

Repository Script Builder

VisualWorks 8.3 includes `Store-RepositoryCreation`, a new parcel containing an improved Store repository script builder. It can be launched by loading the parcel and executing the following:

```
Store.DbRegistry createInstallScript
```

The biggest improvement this script provides is the ability to `GRANT` Store repository permissions to a specified `ROLE` on databases that support users and roles. Previously, these permissions where granted to the `PUBLIC` role or group, allowing Store access to any user able to connect to the database.

Web Automation and JavaScript Evaluation

In VisualWorks 8.3, AppeX includes the `AppeX-JS-Automation` parcel in the `/preview` directory, which provides a framework for the remote evaluation of JavaScript and the automation of browser-based (i.e. web) applications from within Smalltalk.

Evaluating JavaScript from Smalltalk applications

The Web Automation framework can be used to request the evaluation of arbitrary JavaScript from within Smalltalk, and having the results of that evaluation returned to Smalltalk.

The JavaScript evaluation functionality is implemented by the JSEvaluator and JSEvaluatorClient classes. Class JSEvaluator uses a web browser on the server machine to evaluate JavaScript. The automation package contains some examples to illustrate how this capability might be used.

To use the examples, start the JSEvaluator Server, either from the Web Configuration tool or by executing the following in a Workspace:

```
(Server id: 'JSEvaluator') start.
```

For example, to try out JavaScript formatting using the open source library jsbeautifier:

1. Make a change to a JavaScript method in the VisualWorks System Browser.
2. Try **Menu > Edit > Format** to observe the results.

Here are a few expressions that you might try in a Workspace, comparing the result when the JavaScript is evaluated in a browser console:

```
JSEvaluator evaluateJavaScript:  '"a" + "c"'. "ac"
JSEvaluator evaluateJavaScript:  ' 1 2 '. "Error"
JSEvaluator evaluateJavaScript: 'JSON.stringify(window.location);'.
```

Exercising Web Sites Programmatically

The Web Automation functionality also enables you to open and exercise web sites programmatically. This is implemented by the JSAutomator and JSAutomatorClient classes.

The framework injects a test (or automation) script into the javascript downloaded with the application to be tested. The injected JavaScript exercises the web application's functionality and sends results back to the Smalltalk server.

The implementation is as follows.

Class JSAutomator is an application that serves as a proxy between the client (typically, a web browser) and the application to be "automated". As a man-in-the-middle, the JSAutomator can choose to pass on browser requests essentially unchanged to the automated application, modify the requests, or handle the requests entirely. (Refer to the `JSAutomator >> processRequest:` method).

Class `JSAutomatorClient` maintains part of the "injected code" that is sent to the web client. Its JavaScript methods provide functionality to keep track of changes in the browser that are of interest to the tester, and send those results back to the server. (Refer the `JSAtomatorClient >> sendResultStringToServer()` method).

Class `JSAutomatorClient` also provides functionality to simulate user actions on the client, and to modify the behavior of the automated client's own JavaScript code if so desired. (Refer the `clickAnchorWithText()` and `wrapAfter()` methods).

The `JSAutomatorScript` (whose corresponding JavaScript code is also downloaded to the client), kicks off the sequence of testing actions by instantiating instances of the `JSAutomatorClient` and of the user-created (application-specific) subclass of `JSAutomatorClient`.

At present, the web automation framework is used internally by Cincom for regression testing various AppeX example web applications. In a future release, we intend to repackage these functional tests will be repackaged, making them publicly available as examples in their own right.

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 8.3. 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`.

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 (`MatriX.pcl`) is a direct descendant of Polycephaly 1 (`Polycephaly.pcl`), however there are some important 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 MatriX 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.

Chapter

4

Deprecated Features

Topics

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, to facilitate simple loading of deprecated components.

Net Clients

Support for NTLM authentication in HTTP

Since Microsoft stopped supporting NTLM authentication some time ago, we have deprecated NTLM support. All code related to NTLM authentication has been moved to the Obsolete-HTTP parcel in the /obsolete directory.

Microsoft describes the rationale as follows: "Implementers should be aware that NTLM does not support any recent cryptographic methods, such as AES or SHA-256. It uses cyclic redundancy check (CRC) or message digest algorithms (RFC 1321) for integrity, and it uses RC4 for encryption. Deriving a key from a password is as specified in RFC1320 and FIPS46-2. Therefore, applications are generally advised not to use NTLM.".

URISupport methods deprecated

All deprecated methods from the URISupport parcel have been moved to the Obsolete-URISupport parcel, in the /obsolete directory. Load this parcel if your application needs these methods, but consider reworking your application code to use the supported APIs.

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.