VAST Platform 2023
12.0.1

Migration Guide

Tools, tips, and techniques for upgrading from IBM® VisualAge® Smalltalk
and previous versions of the VAST Platform

instantiations

Table of Contents

Migration Guide....22
....22
....22
Migration Guide....23
Preface....24
Notices....25
Trademarks....25
About this book....26
Who this book is for....26
What's new?....26
VAST Platform 2023 (version 12.0.0)....26
Conventions used in this book....26
Tell us what you think....28
Introduction to Migration....29
Preparing for Migration....30
Merge applications from the existing library to the new library....30
Merge the new library into your existing library....31
Migrating from VisualAge Smalltalk V3.0 or earlier....32
Obsolete Code....32
Obsolete VisusalAge code....32
Morphing Tables....33
Obsolete IBM Smalltalk code....34
Migrating applications from VisualAge Smalltalk Standard....34
Generating runtime code....34
Using the migration tool....35
Running the tool....35
Migrating database applications....35

Loading database migration support....36
Database Manager....36
Migrating your application....37
Database Access Set Runtime....38
Dynamic to Static....38
Swapper....38
General migration....38
API maintained for compatibility....39
Dumping, loading, and the Common File System subsystem....39
Example: Unloading using CfsFileDescriptor....39
Example: Loading using CfsFileDescriptor....40
Other migration concerns....40
Enabling applications from Version 1.0 or 2.0 for packaging....40
Promoted features of a Container Details View....41
Updating connections....41
References to pool dictionary CwConstants, AbtCwConstants, or EwConstants....41
AbtNormalGraphic constant used in nonvisual parts....42
Web Connection....43
Extensions of Decimal class....43
Applications that use OSObject subclasses....43
Migrating OSObject subclasses....44
Migrating code that instantiates OSObject directly....44
View wrappers that use attribute-to-attribute connections....45
Migrating the default font for Report Writer reports....46
Packager warning 'No implementors of fixupNlsPool'....47
EwIconTree migration may cause walkback....47
Solution....48
Shortcut keys eliminated from Buttons....49

Migrating from Version 4.0, 4.01, or 4.02 ... 50
Packager changes ... 50
Symbol and class reference adjustment API ... 50
Priority for packaging rules ... 50
Problem and rule policies ... 51
Problem policies ... 51
Rule policies ... 51
Migrating distributed applications ... 52
Distributed Smalltalk infrastructure ... 52
Object space ... 52
Description ... 52
SST equivalent ... 53
Communications ... 53
Description ... 53
SST Equivalent ... 53
Marshaling ... 53
Description ... 53
SST equivalent ... 53
Logical process ... 54
Description ... 54
SST equivalent ... 54
Activation ... 54
Description ... 54
SST equivalent ... 54
Security ... 54
Description ... 54
SST equivalent ... 54
Distributed development tools ... 55

Remote Workspace....55
Description....55
SST equivalent....55
Remote Transcript....55
Description....55
SST equivalent....55
Remote file dialog....55
Description....55
SST equivalent....56
Distributed Inspector....56
Description....56
SST equivalent....56
Distributed Debugger....56
Description....56
SST equivalent....56
Distributed garbage collection....56
Description....56
SST equivalent....56
Distributed load....57
Description....57
SST equivalent....57
Name server....57
Description....57
SST equivalent....58
Event profiling (method calls)....58
Description....58
SST equivalent....58
VA Smalltalk Parts....58

Description..... 58
SST equivalent..... 58
Distribution menu..... 59
Description..... 59
SST equivalent..... 59
Distributed runtime APIs..... 59
Basic remote object API..... 59
Description..... 59
SST equivalent..... 59
Porting guidelines..... 60
Object copying API..... 60
Description..... 60
SST equivalent..... 60
Porting guidelines..... 61
Parallel processing API..... 61
Description..... 61
SST equivalent..... 62
Porting guidelines..... 62
Name server..... 62
Description..... 62
SST equivalent..... 62
Porting guidelines..... 63
Event callbacks (connect/disconnect)..... 63
Description..... 63
SST equivalent..... 63
Porting guidelines..... 64
Distributed exceptions..... 64
Description..... 64

SST equivalent..... 64
Porting guidelines..... 65
Object space management..... 65
Description..... 65
SST equivalent..... 65
Porting guidelines..... 66
Distributed runtime APIs..... 67
DS-API..... 67
DT-API..... 68
Other migration concerns..... 68
Using VA Smalltalk features in server applications..... 68
Runtime code for nonvisual parts in server applications..... 68
Headless runtime applications..... 69
Signalling events with arguments..... 69
Ending server applications..... 70
MVS applications that reference subsystem type 'TM'..... 70
Smalltalk error message in a CICS workstation environment..... 71
Search path for multiple images of Smalltalk using CICS classes..... 71
Snapshot files..... 71
Migrating from Version 4.5..... 72
Formatted Text Field part..... 72
Enhanced shared library support on AIX..... 73
Server applications using .mpr files..... 73
Migrating from Version 5.0..... 75
Some base features no longer available..... 75
Some separately priced features no longer available..... 75
Hiding Java packages and projects in browsers..... 75
Providing a customized splash screen..... 76

Migrating control files..... 76
Importing feature code..... 79
Migrating method mappings to JDK 1.2.....80
Changes in Server Workbench..... 80
Current code page support is inconsistent on supported platforms.....80
AbtConnectionSpec >>#targetCodePage: expects a String argument..... 81
Migrating from Version 5.5..... 82
Elimination of required map links in configuration maps.....82
Low-level configuration map reorganization.....82
Time class shape changed..... 83
Default XML serialization methods added for Date and Time..... 83
Changed behavior of millisecondClockValue on OS/390.....84
Behavior change in handling negative system time.....84
Behavior change in creating FileStreams..... 84
Packager 'do not reduce' rule behavior changed.....84
EMSRV startup parameter changes.....86
SST HTTP Servlet API changed.....86
POSIX Migration on OS/390..... 86
Behavior changes for Application Builder parts..... 87
Migrating from Version 6.0..... 88
AbtPortableNotebookPageView inconsistent behavior.....88
ScaledDecimal>>#hash changed..... 89
Subtle change to Compiler.....89
Using ‘Class Variable Initializer’ packaging rules for IC packaging..... 90
Removed unused method #abtXmlPrintOn:namespaces:.....90
Suppress serialization of nil attribute values when attribute mapping specifies Required=”false”..... 91
Migrating from Version 7.0..... 92
Windows theme support interferes with some testing tools.....92

Change..... 92
Action Required..... 92
SST HTTP Server access log..... 92
Migrating from Version 7.5..... 93
Modifiable splash screen on Windows..... 93
Consolidate EtToolsVendorExtensionsApp and VisualAgeVendorExtensionsApp..... 93
Server Smalltalk (SST) feature definition changes..... 93
Merge ENVY/Image Controls apps into other maps..... 94
Change..... 95
Action required..... 95
Changes to the AbtTimestamp class..... 95
Reason for change..... 95
Change..... 95
Action required..... 95
New testing and conversion CLDT API methods..... 95
Manifest file changes some GUI appearance on Windows..... 96
Merge AbtCwControlsEdit and AbtCwControlsRun apps into other maps..... 96
Change..... 97
Action required..... 97
Change to SstHttpServletRequest>>#getContentType..... 97
Reason for change..... 97
Change..... 97
Action required..... 97
Migrating from Version 8.0..... 98
Changes to the UNIXProcess class..... 98
Reason for change..... 98
Change..... 98
Action required..... 98

Change to default RMI stubProtocolVersion.... 98
Reason for the change.... 98
Change.... 99
Action required.... 99
Change to exception handling.... 99
Reason for the change.... 99
Change.... 99
Action required.... 99
Change to Dictionary>>#copy behavior.... 99
Reason for the change.... 99
Change.... 100
Action required.... 100
AbtWaitApp moved from IBM Smalltalk, ALL - UI Run to IBM Smalltalk, ALL - Headless configuration map.... 100
Reason for the change.... 100
Change.... 100
Action required.... 100
New testing CLDT API methods.... 100
Object>>#initialize method added.... 101
Reason for change.... 101
Change.... 101
Action required.... 101
Migrating from Version 8.0.1.... 102
Grease Core and Grease Core Tests maps are obsolete.... 102
Reason for change.... 102
Change.... 102
Action required.... 102
EsString>>#subStrings: changed to be ANSI-compliant and portable.... 102
Reason for the change.... 103

Change..... 103
Action required..... 103
Random class and methods referring to it have been removed..... 104
Reason for change..... 104
Change..... 104
Action required..... 104
SequenceableCollection>>#beginsWith: and #endsWith: have been replaced..... 105
Reason for change..... 105
Change..... 105
Action required..... 105
Duration>>#milliseconds changed to be portable..... 106
Reason for change..... 106
Change..... 106
Action required..... 106
IdentitySet class added as replacement for EsIdentitySet..... 106
Reason for change..... 106
Change..... 106
Action required..... 106
Color class promoted from WbKernel to Kernel..... 107
Reason for change..... 107
Change..... 107
Action required..... 107
SortedCollection private methods #sorted and #sorted: renamed..... 107
Reason for change..... 107
Change..... 107
Action required..... 107
Migrating from Version 8.0.2..... 108
Object>>#value added to CLDT..... 108

Reason for change....108
Change....108
Action required....108
Icon management classes have been reworked....108
Reason for change....109
Change....109
Action required....109
Mastering ENVY/Developer Configuration Expression Prompter configuration map removed....110
Reason for change....110
Change....110
Action required....110
Changes to ScaledDecimal>>#printOn: and #printString....110
Reason for change....110
Change....110
Action required....110
Changes to TrailBlazer Browsers....111
Reason for change....111
Change....111
Action required....111
Migrating from Version 8.0.3....112
Consolidation of EsbBenchmarksES subapplication....112
Reason for change....112
Change....112
Action required....112
Change Float and Scaled Decimal 32-bit hash for better distribution....112
Reason for change....112
Change....112
Action required....113

INI file changes introduced by Preference Settings Framework..... 113
Reason for change..... 113
Change..... 113
Action required..... 114
Change to CgScreen class>>bitmapPath and Locale class>>nlsPath..... 114
Reason for change..... 114
Change..... 114
Action required..... 114
Migrating from Version 8.5..... 116
Changed specification of ODBC dll or shared object in .INI file..... 116
Reason for change..... 116
Change..... 116
Action required..... 116
Changed PlatformConstants::Error to PlatformConstants::Errorregion for Windows..... 116
Reason for change..... 117
Change..... 117
Action required..... 117
Added 2 Parameters to the socketAppender Setting in the ini File..... 117
Reason for change..... 117
Change..... 117
Action required..... 117
Changed default settings for VA Assist Pro..... 118
Reason for change..... 118
Change..... 118
Action required..... 118
Abt Hover Tooltip text now left-aligned by default..... 118
Reason for change..... 118
Change..... 118

Action required..... 119
Class definition template changed in standard browsers.....119
Reason for change.....119
Change..... 120
Action required..... 120
Migrating from Version 8.5.1..... 121
Parameters in the [log4s] stanza of the .INI file should not be enclosed in parentheses..... 121
Reason for change.....121
Change..... 121
Action required..... 121
Time Zone Support Introduced..... 122
Reason for change.....122
Change..... 122
Action required..... 122
Prerequisite ICs changed in many IC packaging instructions.....122
Reason for change.....122
Change..... 122
Action required..... 122
String>>asInteger has been removed from EsLoggingFrameWorkApp.....123
Reason for change.....123
Change..... 123
Action required..... 123
Migrating from Version 8.5.2..... 124
EsLogManager class>>error: and EsLogManager class>>info: have been removed from EsLoggingFrameWorkApp.....124
Reason for change.....124
Change..... 124
Action required..... 124
Signal>>#description changed for compatibility with class-based Exceptions..... 124

Reason for change....125
Change....125
Examples....125
Action required....126
EsTimeZone class has changed....126
Reason for change....126
Change....126
Action required....126
Migrating from Version 8.6....127
Deprecate and replace classes AbtBase64Coder and SstBase64Coder....127
Reason for change....127
Change....127
Action required....128
Encoding....128
Decoding....128
Other....128
Deprecate and replace incorrect Windows code page conversion API wrapper methods....128
Reason for change....129
Change....129
Action required....130
Object>>#triggerEvent: added to AbtCLDTAdditions for portability....130
Reason for change....130
Change....130
Action required....130
Print methods added to Integer and Float in CLDT for portability....130
Reason for change....130
Change....130
Action required....130

Migrating from Version 8.6.1 132
Deprecate classes OSHostent, OSProtoent, and OSServent 132
Reason for change 132
Change 132
Action required 133
EmCompressor compression/decompression algorithm changed 133
Reason for change 133
Change 133
Action required 133
Detailed Description 133
Move SequenceableCollection>>#swap:with: to CLDT for shared use by RBBrowserUIApp and SeasideCoreApp 134
Reason for change 134
Change 134
Action required 135
Dictionary now implements #= and #hash 135
Reason for change 135
Change 135
Action required 135
Multipart Forms Processing Now Removes Only the Trailing Cr/Lf Leaving Whitespace Intact At the End of the Content 136
Reason for change 136
Change 136
Action required 136
SSL Version 2 & 3 Protocols Have Been Deprecated 136
Reason for change 136
Change 136
Action required 136
OpenSSL Libraries are no longer distributed with VA Smalltalk 136

Reason for change....136
Change....137
Action required....137
Migrating from Version 8.6.2....138
Changed representation of socket address in OSSocketAddrInet and SciSocketAddress....138
Reason for change....138
Change....138
Action required....138
Cryptography methods modified to support OpenSSL 1.1....138
Reason for change....139
Change....139
Action required....139
Migrating from Version 8.6.3....140
PlatformFunction and EsEntryPoint parameter/return type accuracy requirement changed....140
Reason for change....140
Action required....140
PlatformFunction and EsEntryPoint type #handle should not be used....141
Reason for change....141
Action required....141
Restrictions on nested use of #lockThreadIn:....141
Reason for change....141
Action required....141
Migrating from Version 9.0....142
AbtWaitApp changes....142
Reason for change....142
Action required....142
EsCompressionStreamsApp changes....143
Reason for change....143

Action required..... 143
ZipWriteStream.....143
ZipReadStream..... 144
Deflate/Gzip Streams.....144
EswStreamExtensions changes..... 145
Reason for change.....145
Action required..... 145
Async Queue Overflow default policy change..... 145
Reason for change.....145
Action required..... 146
Character conversion default policy change on Unix.....146
Reason for change.....146
Action required..... 146
Stream nextLine changes.....146
Reason for change.....146
Action required..... 147
Migrating from Version 9.1..... 148
EmAttachments changes.....148
Reason for change.....148
Action required..... 148
SstImap4Communications changes..... 148
Reason for change.....149
Action required..... 149
SstSMTPCommunications changes..... 149
Reason for change.....149
Action required..... 149
NeoJSON WriteStream extension changes..... 149
Reason for change.....149

Action required..... 150
Manifest file changes some GUI appearance on Windows..... 150
Reason for change..... 150
Action required..... 150
How do I disable HiDPI?..... 150
General database changes related storing handles..... 151
Reason for change..... 151
Action required..... 151
Server Runtime's (SRT's) are now self contained..... 152
Reason for change..... 152
Action required..... 152
HiDPI impacts class side OSWidget level extent methods..... 153
Reason for change..... 153
Action required..... 153
HiDPI impacts fonts..... 153
Reason for change..... 153
Action required..... 153
Deprecate AbtConditionalWait>>#setDefaultWaitValues..... 154
Reason for change..... 154
Action required..... 154
Migrating from Version 9.2..... 155
Transcript search results sorted..... 155
Reason for change..... 155
Action required..... 155
HiDPI requires default font change..... 155
Reason for change..... 156
Action required..... 156
Migrating from Version 9.2.2..... 157

Non-Inheritable Handles by default on Windows..... 157
Reason for change..... 157
Action required..... 157
Default OpenSSL PlatformLibrary Name Mapping Update..... 157
Reason for change..... 158
Action required..... 158
Exponential notation will now always answer a Float instance..... 158
Reason for change..... 158
Action required..... 158
Migrating from Version 10.0.0..... 159
EsCodePageUtilitiesApp moved to CodePageSupport..... 159
Reason for change..... 159
Action required..... 159
Object>>#abrAsClass has been deprecated..... 159
Reason for change..... 159
Action required..... 160
Migrating from Version 11.0.0..... 161
"Rectangle" key renamed to "RectangleFunction" in #_PRAGMA_PlatformFunctions..... 161
Reason for change..... 161
Action required..... 161
Migrating from Version 11.0.1..... 162
#sstBase64Encoded now uses Base64Coder instead of Base64CoderMime..... 162
Reason for change..... 162
Action required..... 162
On Linux, glibc function ftime() usage in VM has been removed..... 162
Reason for change..... 162
Action required..... 163
LDAP search parameter handling changed..... 163

Reason for change....163
Action required....163
Migrating from Version 12.0.0....164
CwTable>>#fullRowSelect has been deprecated....164
Reason for change....164
Action required....164
CwTable>>#editLabels has been deprecated....164
Reason for change....164
Action required....164
Instantiation of readers and writers for NeoCSV and NeoJSON has changed....165
Reason for change....165
Action required....165
SstHttpClient>>#transportScheme: has been deprecated....165
Reason for change....165
Action required....165
Deprecated Classes and Methods....167
Migration Tools....171
Library Importer Tool....171
Logging....173
Export/Import Passive Image Properties....173

Migration Guide

Version 12.0.1
Copyright 2023

Migration Guide

This book covers upgrading existing VAST (VA Smalltalk) applications to the most current version of VAST Platform (VA Smalltalk). You should review this book prior to loading your existing applications into the new library of code.

Preface

Version 12.0.1, 2023

This edition applies to Version 12.0.1 of the VAST Platform (VA Smalltalk) products, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product. The term "VA Smalltalk," as used in this publication, refers to the VA Smalltalk product set.

Note:
Before using this document, read the general information under Notices.

If you have comments about the VAST Platform (VA Smalltalk) or this document, please contact us using a method on: Instantiations “contact-us” page

The online version of this document has an email button ( ) at the top right of any page to email your comments to us. You can also email your comments to us at vast-support@instantiations.com

When you send information to Instantiations, you grant Instantiations a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright Instantiations, Inc. 2005, 2023. All Rights Reserved

© Copyright International Business Machines Corporation 1994, 2002. All rights reserved.

Notices

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY, USA 10594.

Instantiations may change this publication, the product described herein, or both without notice.

Disclaimer:

The screen shots in this document will not exactly match what appears in the product. Differences will be slight and will not impair use of the document.

Trademarks

‘Instantiations’ and the ‘intersecting circle design’ are registered trademarks of Instantiations, Inc. in the United States. All product names, trademarks, and registered trademarks are property of their respective owners. Company, product, and service names not owned by Instantiations are used for identification purposes only. Use of these names, trademarks, and brands does not imply endorsement.

About this book

This book covers upgrading your existing applications from IBM VisualAge Smalltalk Enterprise Version 3.0 through 6.0.4, VA Smalltalk Versions 7.0 through 9.2.2, and VAST Platform Verstion 10.0.0 and later to the current VAST Platform (VA Smalltalk) version. You should review and understand the following material, as appropriate, prior to loading your existing applications into the current version of VAST Platform (VA Smalltalk.)

To a limited extent, this book also covers how to upgrade IBM VisualAge for Smalltalk Versions 1.0 and Version 2.0 applications to the current version of VAST Platform (VA Smalltalk.)

Who this book is for

This book is for developers of applications built using Versions 1.0 through Version 6.0.4 of IBM VisualAge Smalltalk Enterprise, and all versions of VA Smalltalk prior to the current version, who now want to work on their previously built applications using the current version of VA Smalltalk.

If you have never used IBM VisualAge Smalltalk Enterprise or VA Smalltalk before, this book will be of little interest to you.

What's new?

VAST Platform 2023 (version 12.0.1)

The following new or changed items are covered in this edition:

- Migration from VA Smalltalk V7.0
- Migration from VA Smalltalk V7.5
- Migration from VA Smalltalk V8.0, V8.0.1, V8.0.2, and V8.0.3
- Migration from VA Smalltalk V8.5 and V8.5.1
- Migration from VA Smalltalk V8.6, V8.6.1, V8.6.2 and V8.6.3
- Migration from VA Smalltalk V9.0, V9.1, V9.2 and 9.2.2
- Migration from VAST Platform V10.0.0
- Migration from VAST Platform V 11.0.0 and V11.0.1
- Migration from VAST Platform V12.0.0
- Migration Tools
- Deprecated Classes and Methods

In addition, the chapters of the book have been reordered so you can read the book more naturally from front to back.

Conventions used in this book

This book uses several conventions that you might not have seen in other product manuals.

Throughout this book, the term "Windows" applies generically to all versions of Windows supported by Smalltalk. Information applicable to a specific Windows environment is noted accordingly.

Tips and environment-specific information are flagged with icons:

Shortcut techniques and other tips

FOR WINDOWS:

VAST Platform (VA Smalltalk) for Windows

FOR LINUX (UNIX):

VAST Platform (VA Smalltalk) for UNIX platforms

These highlighting conventions are used in the text:

Boldface

Used for	Example
New terms the first time they are used	VAST Plaform uses **construction from parts** to develop software by assembling and connecting reusable components called **parts**.
Items you can select, such as push buttons and menu choices	Select **Add Part** from the **Options** pull-down. Type the part's class and select **OK**.

Italics

Used for	Example
Special emphasis	Do *not* save the image.
Titles of publications	Refer to the *Installation Guide*.
Text that the product displays	The status area displays *Category: Data Entry*.
VAST Plaform programming objects, such as *attributes, actions, events, composite parts,* and *script names*	Connect the window's *aboutToOpenWidget* event to the *initializeWhereClause* script.

`Monospace font`

Used for	Example

`VAST Plaform scripts and other examples of Smalltalk code`	`doSomething`
	`| aNumber aString |`
	`aNumber := 5 * 10.`
	`aString := 'abc'.`

`Text you can enter`	`For the customer name, type John Doe`

Tell us what you think

Please take a few moments to tell us what you think about this book. The only way for us to know whether you are satisfied with our books or how we can improve their quality is through feedback from customers like you.

You can email comments to vast-support@instantiations.com or post comments and technical questions to the VAST Community Forum. A link to the Forum can be found at "Support Forum" or from any of the VAST Platform Help menus. Instructions on what information to include with technical problems can be found at "Technical Support" on our VA Smalltalk web page. Or you can go to the "Contact Us" page of our website.

The online version of this document has an e-mail button ( ) on each page to enable you to give us your feedback.

Introduction to Migration

When you install a major release of VAST Platform / VA Smalltalk, such as Version 10.0.0, Version 7.5 or Version 8.0, into the default directory location, any older installations of VAST Platform / VA Smalltalk or VisualAge Smalltalk are not affected. This means that you can run the new version and the old version side-by-side while you do your migration activities.

Following the successful installation of your new version of the product, you should follow these steps to migrate your application to the new development environment:

- Backup your current development library.
- Save to files any text from open workspaces or from the Transcript window of your current development image.
- Merge your current development library and the new development library. You can either merge applications from your existing library into the new library or merge the new library into your existing library. No matter which direction you perform the merge, the merge operation only copies versioned maps and applications.

  If you merge your existing library into the new library, you will leave all open editions and scratch editions of applications and maps behind -- if you do this, you should archive your existing library in case you need to refer to the information that wasn't copied into the new library.

  If you wish to retain all open editions of maps and applications, you can merge the entire new library into your current library. This has the advantage of keeping all your work in one place, but it keeps all code, even code you may wish to discard. If the library becomes too large, you can clone it after you are satisfied with the migration and discard all open editions at that time.
- For each product version there are specific migration concerns. Each product version is covered in a separate chapter. You should start reading at the chapter describing the product version you are migrating from, and continue reading to the end of this book to understand the complete migration picture. You can perform the migration activities as you go.
- If you have changed any of the product applications, you should evaluate your changes and re-merge and test your changes in the new version of the product.
- Run any regression tests you have for your application against the new configuration.
- Once you are satisfied that the application performs as specified in the development image, you should package your application and execute your runtime tests.

Preparing for Migration

VAST Platform (VA Smalltalk) code is held in a **code repository**, also referred to as a **manager library** or **development library**. If you are currently using VisualAge Smalltalk or VA Smalltalk or a previous version of of VAST Platform, then after the successful installation of your new version of VAST Platform (VA Smalltalk) you will have at least 2 development libraries on your system. In the following discussion, we will refer to the development library you have been using as the *existing development library* and the development library installed with the current version of VAST Platform as the *new development library*.

To prepare for migration, you need to merge these two libraries. You can either:

1. merge your applications from the existing development library into the new development library or
2. merge all applications from the new development library into the existing development library

Both mechanisms are equally valid. Which is right for you depends on your goals.

If you import the new development library into your existing larger library, you can keep all your open editions of methods, classes, applications, and configuration maps.

On the other hand, the library can often fill up with purged, scratch, open editions from over the many years of working with the manager. Migrating to a new version could be a grood opportunity to reduce the size of the manager (sometimes significantly). In this case, you can take advantage of the fact that since the Library Importer Tool will only carry over versioned items and choose to import your existing larger library into a clean new development library.

Whatever you do, make a backup of your existing development library in case you want to explore the other direction.

Merge applications from the existing library to the new library

This section covers how to merge applications developed using earlier versions of VisualAge Smalltalk or VA Smalltalk or VAST Platform (VA Smalltalk) to the current VAST Platform library by importing them from the existing library.

Using the browsers in your current development image, do the following:

1. Version and release all open editions of classes and subapplications, and version the applications.
2. Save to files any text from open workspaces or from the Transcript window that you want to keep.

Then, using the browsers in your new development image, you can import your work from the earlier version's development library into the new development library. The *Smalltalk User Guide* explains how to version and import applications and configuration maps.

To quickly move your applications, you might put them into one or more configuration maps while working in an image from an earlier release, and then, while working in the newly installed product image, import the configuration maps into the newly installed library.

Merge the new library into your existing library

This section covers how to copy product code from a newly installed VAST Platform (VA Smalltalk) library to your existing VisualAge Smalltalk or VA Smalltalk or VAST Platform library. The steps in this section describe a simple process for migrating to the current version of VA Smalltalk without having to identify and import all of your work.

When you install the current version of VAST Platform (VA Smalltalk)Library Manager, the installation process creates a new library (for example, mgr1201`.dat`). Merging code from the current version of VAST Platform (VA Smalltalk) into your existing library involves running the Library Importer Tool to import the newer product code into your library. You can then resume work using your current VAST Platform (VA Smalltalk) image. See Library Importer Tool for more information.

Migrating from VisualAge Smalltalk V3.0 or earlier

This chapter addresses concerns for users of IBM VisualAge Smalltalk, or IBM Smalltalk, Version 3.0 or earlier who are migrating to the current version of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Obsolete Code
- Moving applications from VisualAge Smalltalk Standard
- Using the migration tool
- Migrating database applications
- Swapper
- Other migration concerns

Obsolete Code

Some classes (or class names), or methods in those classes, are no longer supported. to load or run your Version 3.0 application s that use obsolete code for Version 4.5, do the following:

1. Look through the list of obsolete code below. Indentify those obsolete classes used by your Version 3.0 applications.
2. Examine the configurations maps Visual Age, Obsolete Parts for classes extended widgets.
3. Determine which applications in these maps contain obsolete classes used by your Version 3.0 applications.
4. From the maps, load the pertinent applications.
5. Try loading your Version 3.0 applications again.

Some of the classes listed below are application classes. If your Version 3.0 application's direct prerequisites include any of the classes below, load the appropriate configuration map and fix your applications so they no longer use obsolete code. Not that this only applies for direct prerequisites, and not for prerequisites well up the tree of prerequisites.

Obsolete VisusalAge code

The following classes are themselves obsolete or contain code that is obsolete:

- *AbtBuildInterfacesApp*
- *AbtBuildPartsApp*

- *AbtBuildviewsApp*
- *AbtEditBaseApp*
- *AbtExternalObjectsApp*
- *AbtTableColumnEditPart*
- *AbtTableColumnEditPartInputPolicy*
- *AbtTableColumnVisualPolicy*
- *AbtTableEditPart*
- *AbtTableColumnView*
- *AbtTableUseCompositePartAsVisualPolicy*
- *AbtTableView*
- *AbtMRIManager*
- *AbtNLSCoordinator*
- *AbtNLSStringMapper*
- *AbtTableColumnView*
- *AbtTableTpoTextConverterAdapter*
- *AbtPartPackagerApp*
- *AbtPersistencyBaseApp*
- *AbpPOMObjectExtractor*
- *AbtTableCell*
- *AbtTableCellManager*
- *AbtTableColumn*
- *AbtTableColumnLebel*
- *AbtTableColumnManager*
- *AbtTableColumnSizer*
- *AbtTableCornerLabel*
- *AbtTableGeometry*
- *AbtTableGridDrawer*
- *AbtTableLabel*
- *AbtTableLabelManager*
- *AbtTableMinimalConverterAdapter*
- *AbtTableRowLabel*
- *AbtTableScrollManager*
- *AbtTableSelectionCallbackData*
- *AbtTableSelectionPolicy*
- *AbtTableSingleSelectionPolicy*
- *AbtTableValueCallbackData*
- *AbtTableWidget*
- *AbtTableWidgetApp*
- *SequenceableCollection*

If your application uses these classes, load applications in the configuration map *VisualAge, ObsoleteParts* that contain obsolete classes used by your Version 3.0 application. Further, if your application uses *AbtTableView*, follow the step in "Morphing Tables" after loading the applications.

Morphing Tables

If your Version 3.0 application uses *AbtTableView*, you will need to morph the table so it uses the Version 6.0 *AbtContainerDetailsView* as follows:

1. If you have not done so already, load applications in the configuration map *VisualAge, ObsoleteParts* that contain obsolete classes used by your Version 3.0 application.
2. Load your Version 3.0 application
3. Open the visual part that contains the table.
4. Press mouse button 2 on the table and select **Morph** into from its popup menu.

5. Select *AbtContainerDetaislView*
6. Save, version, and release your work.
7. Unload the applications loaded from the configuration map *VisualAge, Obsolete Parts*.

Obsolete IBM Smalltalk code

The following classes are themselves obsolete or contain code that is obsolete:

- *EwBase*
- *EwContainerDragAndDropSupport*
- *EwContainerSupport*
- *EwDragAndDropSupport*
- *EwDrawnListSupport*
- *EwListSupport*
- *EwNotebookSupport*
- *EwProgressIndicatorSupport*
- *EwSliderSupport*
- *EwSpinButtonSupport*
- *EwSplitWindowSupport*
- *EwrSupport*
- *EwToolBarSupport*

If your application uses these classes, load *ENVY/Image Ew R3.0 Compatibility*.

Migrating applications from VisualAge Smalltalk Standard

If you developed applications using VisualAge Smalltalk Standard, you must complete the following steps to migrate your code to the current VA Smalltalk library.

1. Generate archival code for your applications into new applications. Refer to Version 3.0 or earlier documentation to learn how to generate archival code.
2. File out the original (runtime) applications and the new archival applications.

   Again, refer to Version 3.0 or earlier documentation for information on filing out code.
3. File the applications into a current VA Smalltalk development image. The chapter "Managing your VA Smalltalk application" in the *Visual Programming User Guide* explains how to file the code into a VA Smalltalk development image.
4. Generate runtime code for the applications as described in Generating runtime code.

The *Smalltalk User Guide* provides extensive information on versioning and releasing.

Generating runtime code

If you used VisualAge Smalltalk Standard, you must generate runtime code for all applications that you migrate to the current version of VA Smalltalk.

To accomplish this, select your application in the Organizer window and select **Generate > Runtime Code** from the **Applications** menu.

Note:

You can also generate runtime code by opening the Composition Editor on individual part classes of your application and selecting **Save Part** from the **File** menu. Beginning with Version 6.0, VisualAge Smalltalk automatically generates runtime code every time you save your parts. For migration, however, the first method described is likely much faster.

Generated runtime code includes the private instance method named *abtBuildInternals*, which contains all of the information needed to recreate the visual subparts and connections of your part at runtime, as you defined them in the Composition Editor. It also includes the private class methods beginning with *IS_* (short for interface specification), which contain all information needed to re-create the features of your part at runtime, as you defined them in the Public Interface Editor.

Using the migration tool

When you load Version 3.0 applications into a current VA Smalltalk development image, warning messages such as "Defined global <name> in unmanaged namespace" might print to the System Transcript window. These messages indicate that you need to change your code so it runs cleanly on the current version of VA Smalltalk. The base migration tool can help you fix the code.

If you are migrating database applications, see Migrating database applications.

Running the tool

When a current VA Smalltalk development image has the *AbtMigrationApp* application loaded, the **Migrate > Base Migration** menu choice is available from the **Applications** menu of the Organizer. This provides a tool for migrating your work to the current version of VA Smalltalk.

To run the migration tool, select your application(s) in the Organizer and then select **Migrate > Base Migration**.

While the tool runs a Migration Information workspace opens, indicating what classes migrated successfully. Note the suggestion printed to the workspace when the tool finishes running; it advises you to version and release migrated applications and classes and, if necessary, regenerate archival code.

When you finish migrating your work, you can unload the *AbtMigrationApp* application if you wish.

Migrating database applications

You have three options for migrating work that uses a database feature:

Database Manager menu choice
Use this option to migrate the following:

- Native DB2 support to CLI
- The database manager IBM DATABASE2 has been discontinued and replaced by IBM DB2-CLI. If you have connection specs which use IBM DATABASE2, you must complete this step for migrating native DB2 support to CLI.
- Multi-Database applications to ODBC support (or for ORACLE, to native ORACLE support)
- An application to a new manager if you changed the connection alias so it points to a different database manager.

Database Access Set Runtime menu choice
Use this option to add universal fields to your database application. This is required for all Version 3.0 database applications.

Dynamic to Static menu choice
Use this option to migrate a DB2 CLI application with universal fields to a Static DB2 application.

The option(s) you choose, depend on the database your Version 3.0 application supports and the database you want your current VA Smalltalk application to support. You may need to use more than one option. For example, if you are migrating a Version 3.0 native DATABASE2 application to Static DB2, use all three options sequentially:

1. Use the **Migrate > Database Manager** choice of the Organizer's **Applications** menu to migrate native DATABASE2 code to DB2 CLI.
2. Use the **Migrate > Access Set Runtime** choice of the Organizer's **Applications** menu to add support for universal fields to the DB2 CLI code.
3. Use the **Migrate > Applications** menu of the Organizer. choice of the **Applications** menu to migrate the DB2 CLI code with universal fields to Static DB2.

The materials below describe how to use the three options.

Loading database migration support

Use the **Load/Unload Features** menu choice of the Organizer's **Options** menu to load database migration support into your image.

When you load the **VA: Database, Parts** feature, the **Database Manager** and **Access Set Runtime** menu choices are added to the **Applications** menu of the Organizer. When you load the **VA: Database, Static DB2 Support** feature, the **Dynamic to Static** menu choice is added.

Refer to the *Visual Programming User Guide* for information on loading and unloading features.

Database Manager

Before using the **Database Manager** option, make sure you have installed the database manager to which you want to migrate and, if you are using ODBC, make sure you have installed the proper ODBC drivers. Refer to the *Database Guide* for information on ODBC support drivers.

If you have VisualAge Version 3.0 applications using the native DB2 manager IBM DATABASE2, you will have to load the feature **IBM ST: Database, Obsolete DB2 Support** before loading your applications. After migrating you can unload this feature.

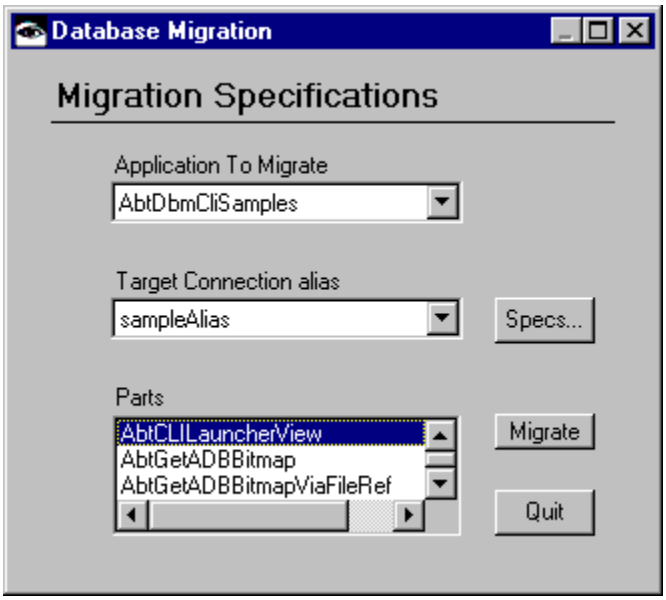
When migrating from one database manager to another or from one database to another, you need to be careful of the following:

- Make sure that the table structures are the same.
- If you use high-level qualifiers, make sure they are the same.
- Make sure that the data types defined in the tables are consistent with the data types defined in your user interface.

Migrating your application

To migrate a database application, follow these steps:

1. Select the application(s).
2. Open the Database Migration window by selecting **Migrate > Database Manager** from the **Applications** menu.
   The window opens as follows.



3. From the **Application To Migrate** field, select the application you want to migrate.
   If you have defined connection specifications for the application, these appear in the **Target Connection alias** field.
4. Select the connection alias you want to use in the application from the **Target Connection alias** field. If your application does not have a connection specification, select the **Specs** button.
   For instructions on creating a connection specification, refer to the *Database Guide*.
5. From the **Parts** field, select the parts that you want to migrate, and then select the **Migrate** button.
6. After the tool runs, select **Quit**.

Database Access Set Runtime

If your Version 3.0 application contains an access set, you must use the **Database Access Set Runtime** option. This option migrates the query specs in the access set to point to the new universal field types. It also prepares the query to be used in a more optimized fashion, improving performance.

To use this option, select the application and then **Migrate > Database Access Set Runtime**.

Dynamic to Static

To turn DB2 CLI queries into precompiled static queries, use the **Dynamic to Static** option:

1. Select the application.
2. Select **Migrate > Dynamic to Static**. This opens the Static Migration window, which shows the selected application highlighted.
   Use the window to detect the DB2 CLI connections defined in the application.
3. Select one or more detected connections. The corresponding package name is shown in the **Package Name** field.
4. Select the **Migrate** push button.

VisualAge evaluates the database parts that use CLI connections. It generates static queries according to the settings in the database parts and places the static queries in the specified package.

Swapper

This section covers migrating code that uses classes and methods in the Swapper. The classes and methods have changed significantly from Version 3.0. For information on the current version of the Swapper, refer to the *Smalltalk User Guide*.

General migration

All uses of the old API for *ByteArray* are now obsolete. They can and should be mapped on top of the adequate stream (memory stream, network stream, and so on). This should also improve performance, because the computation of the size of the *ByteArray* is not necessary. Streams grow as needed.

Unlinking and relinking have been replaced by a more powerful mechanism, replacements. Any object can specify a dumping replacement; another object that will replace it when it is dumped. In turn, any object can also specify a loading replacement; another object that will replace it when it is loaded. This provides two-phase replacement. For instance, an object A that represents a file can be dumped as an intermediate representation B (a dumping replacement for A) which carries the path of the file. Upon load, B specifies a loading replacement C, which is a valid file again, based on the path defined by B.

For backwards compatibility, class names such as *ObjectSwapper*, *ObjectLoader*, *ObjectDumper* have been maintained. The *ObjectSwapper* API in IBM Smalltalk V3.0, although obsolete for most of the

calls, has been implemented to facilitate migration of existing code. Class prefixes have been added for new classes, to minimize the name collision problem. Public classes from previous releases, like *ObjectSwapper*, *ObjectLoader* and *ObjectDumper* do not have prefixes added for compatibility reasons.

API maintained for compatibility

This section contains a list of features still available for compatibility reasons, but which are now considered obsolete. It is strongly recommended that if you have code that relies on any of these features, it be updated as suggested below.

- A size calculation method is provided to determine the amount of disk space the object dump will occupy without unloading it. This operation was used primarily to compute the size of a *ByteArray* for this kind of unloading. Since swapping now uses streams, there is no need to pre-compute sizes. The operation is still provided, but is inefficient.
- Objects may be loaded/unloaded to any OS file descriptor at a specified file offset, or to byte objects (for example, instances of *ByteArray*, instances of *String*). The new release works on top of streams. This old API is implemented on top of file streams, but may not be supported in future releases.
- Objects may be unlinked from the dump using either a class based, instance variable based, or object identity test mechanism. Objects unlinked using the object identity test mechanism may be re-linked to any specified object on load. Unlinking and relinking are implemented with the new mechanism, replacement. If unlinking is all you need, just specify a dumping replacement being *nil* for that particular object.

Dumping, loading, and the Common File System subsystem

For compatibility, the current version of the Swapper contains identical protocols as its previous releases, for dumping to and loading from files by means of the Common File System subsystem. Like in Version 3.0, the following obsolete methods require a different file handle parameter.

Methods in *ObjectDumper*

- *unload:intoFileHandle:atOffset:*
- *unload:intoFileHandle:atOffset:maximumLimit:*
- *unload:intoFileHandle:atOffset:maximumLimit:errorStream:*

Methods in *ObjectLoader*

- *loadFromFileHandle:atOffset:*
- *loadFromFileHandle:atOffset:errorStream:*

Example: Unloading using CfsFileDescriptor

The file handle parameter required by the `ObjectDumper` must be an instance of `CfsFileDescriptor`. The following example shows how one of the methods, `unload:intoFileHandle:atOffset:`, is used.

```
| aCfsFileDescriptor anObject result |

aCfsFileDescriptor := CfsFileDescriptor
```

```
    open: 'someobj.swp'
    oflag: ORDWR | OCREAT.
  aCfsFileDescriptor isCfsError
    ifTrue: [^self error: 'Cannot open file'].
  anObject := 1@2.
  result := ObjectDumper new
    unload: anObject
    intoFileHandle: aCfsFileDescriptor
    atOffset: 0.
  aCfsFileDescriptor close.
  System message: result printString.
```

Example: Loading using CfsFileDescriptor

The file handle parameter required by the `ObjectLoader` must also be an instance of `CfsFileDescriptor`. This example shows how `loadFromFileHandle:atOffset:` is used.

```
  | aCfsFileDescriptor someObject |

  aCfsFileDescriptor := CfsFileDescriptor
    open: 'someobj.swp'
    oflag: ORDONLY.
  aCfsFileDescriptor isCfsError
    ifTrue: [^self error: 'Cannot open file'].
  someObject := ObjectLoader new
  loadFromFileHandle: aCfsFileDescriptor
    atOffset: 0.
  aCfsFileDescriptor close.
  someObject inspect.
```

Other migration concerns

This section describes how to fix an assortment of problems you might encounter when migrating your work.

Enabling applications from Version 1.0 or 2.0 for packaging

If you are migrating a Version 1.0 or Version 2.0 application, you may get a packager warning saying there are "No implementors of fixupNlsPool" when trying to package. If you are packaging as an IC, this will result in a runtime error when an attempt is made to run the packaged application.

To get around this, browse implementors of *initializeAfterLoad* in your image. If there are implementors of this method (generated by VisualAge) in any of your view classes, delete them. This is obsolete code that was generated in previous VisualAge releases that is no longer needed. This should clear up this particular packager warning and your application should run if packaged properly.

Promoted features of a Container Details View

The migration tool does not handle the case where you promoted any of the following action, attributes, or events of a Container Details View:

forcePacketRequest
: Action

packetEnabled
: Attribute or event

totalRows
: Attribute or event

packetRequested
: Event

If, in your Version 3.0 application, you promoted any of these features, embedded a Container Details View into another view, and either made connections to or set the values of any of the above features, you must remove these connections or set values in the view they embedded into.

Updating connections

In Version 3.0, there was an attribute called *result* that you could select when connecting to an existing connection. In later releases, the *result* attribute was divided into two separate attributes called *normalResult* and *errorResult*. This is helpful because the need to do a check to determine if the result is an error is now not necessary. You can now make a connection from either one or both to something that is appropriate for that case. If you make connections from both attributes, only one will fire depending on whether the result was normal or an error.

If you bring in views from Version 3.0 that had the connection from *result*, this connection will still work as always. However, we are recommending that as you come across them, you should change the connection to be from the appropriate attribute.

You may notice when migrating from Version 3.0, if you had a connection from *result* to an action that required a parameter, this connection may now appear dashed. It will still work; however, to cause the line to not be dashed, you will need to open the settings on this connection and change the connection to be from *normalResult* or *errorResult* (whichever is appropriate).

References to pool dictionary CwConstants, AbtCwConstants, or EwConstants

If you are having difficulty loading a Version 3.0 application due to compile errors referring to undefined constants, you may need to do the following.

In Version 3.0, *AbtAppBldrPart* had the following pool dictionaries defined: *CwConstants*, *AbtCwConstants*, and *EwConstants*. If one of your classes subclasses from *AbtAppBldrPart* (like a nonvisual part), these pool dictionaries are in your hierarchy. In later versions there was some refactoring

of Pool Dictionaries. The pool dictionaries *CwConstants*, *AbtCwConstants*, and *EwConstants* were moved and defined in *AbtAppBldrVisual*, *AbtAppBldrView*, and probably in other classes. If your classes subclass directly from *AbtAppBldrNonVisual* or *AbtAppBldrPart*, the pool dictionaries are no longer defined directly there but in the classes mentioned above. Therefore, you must do one of the following:

1. Execute the following code in the Transcript:

   ```
   EmImageBuilder cancelIfMethodsDoNotCompile: false.
   ```

   Then reload. After you reload, you must add an entry in the class definition of your class that previously could not compile for the missing pool dictionary. After you do this, reload the class or application again.
2. Define the needed pool dictionaries in your class before porting to the current version of VA Smalltalk.

AbtNormalGraphic constant used in nonvisual parts

In Version 3.0, for applications containing nonvisual parts that referenced the *AbtNormalGraphic* constant in *AbtConstants*, the *AbtConstants* pool dictionary was inherited from *AbtPart*. The class hierarchy was as follows:

```
AbtPart

  ...

  AbtCompositePart

    ...

    AbtAppBldrPart

      ...

      MyNonVisualPart
```

In VisualAge Smalltalk V4.0, that constant moved from *AbtConstants* in *AbtPart*. The constant is now in the *AbtCWAdditionsConstants* dictionary. That dictionary is in the hierarchy of *AbtAppBldrVisual* but not *AbtAppBldrNonVisual*. So, if you have a nonvisual part that references this constant, when you try to load this class you will receive a compiler error in the Transcript similar to the following:

```
compiler error "undefined"
--> AbtNormalGraphic
Error: 357 Cannot complete the load because CustomerNonVisualPart>>#methodName
(3/26/97 10:35:09 AM) does not compile.
NOTE: If you reload after executing: EmImageBuilder
cancelIfMethodsDoNotCompile: false
methods which do not compile will be deleted.
```

You have several options at this point.

1. If you only use this constant in a method or two, you can execute the code in the Transcript:

   ```
   EmImageBuilder cancelIfMethodsDoNotCompile: false
   ```

Then, reload. After you have reloaded, you will need to add an entry in the class definition of your nonvisual class for the pool dictionary *AbtCWAdditionsConstants*. Once this is done, you will need to reload the class again or reload the methods that reference this constant.

2. If you use this constant a lot, you can change your Version 3.0 classes so they declare an empty pool dictionary entry in the Smalltalk dictionary for *AbtCWAdditionsConstants*, which you can then put in the class definition for your nonvisual parts that reference this constant. Make sure to version and release your changed work while using a Version 3.0. image. This will allow you to load cleanly into current VA Smalltalk image.

Web Connection

After importing your Web Connection applications built with Version 3.0, but before loading them into a current VA Smalltalk image, execute the following:

```
EmImageBuilder cancelIfMethodsDoNotCompile: false
```

This statement removes the class *AbtNormalGraphic*, which is no longer in the product, from *abtBuildInternals* methods in your applications when your applications load. When you later regenerate code for your parts, by selecting **Generate > Runtime Code** from the **Parts** menu, the *abtBuildInternals* methods will be regenerated.

After loading your Web Connection applications, select your applications and then select **Migrate > Base Migration** from the **Applications** menu of the Organizer. Then select **Migrate > Web Connection** from the same menu.

If you do not migrate the applications, the nonvisual Web Connection parts will appear without icons.

Extensions of Decimal class

If your Version 3.0 application contains an extension of *Decimal* class, you will have trouble migrating to the current version of VA Smalltalk and will get a transaction protocol error.

To enable an error-free migration, while in a Version 3.0 image, create a *ScaledDecimal* class in the *DecimalMath* application. Then, extend the *ScaledDecimal* class into your application and move all methods from their *Decimal* extension to the *ScaledDecimal* extension. After you make these changes and version your application, you can import it into a current VA Smalltalk development library.

Ensure that you make the changes to your application in a Version 3.0 image, as making these changes in a current VA Smalltalk image can cause problems with the Swapper.

Applications that use OSObject subclasses

In VisualAge Smalltalk V4.0, the class *OSObject* became an abstract class which should not be directly instantiated or subclassed. An *OSObject* subclass called *OSPtr* now contains the function previously available in *OSObject*. Additionally, *OSPtr* contains a subclass called *AbtForeignOSObject* which

supports code page conversion. Existing applications that currently use *OSObject* subclasses or instances to share data with external functions need to be migrated to use *OSPtr* or another suitable *OSObject* subclass.

Migrating OSObject subclasses

To migrate *OSObject* subclasses, first determine which classes need to be migrated:

1. Load existing applications that use *OSObject* instances or subclasses to interface with external functions.
2. In the System Transcript, select **Browse Hierarchy** from the **Tools** menu. A prompter opens, requesting a class name.
3. Type `OSObject` and select **OK**. A hierarchy browser opens, showing the class *OSObject* and its direct subclasses.
   The classes *OSImmediate*, *OSPtr*, and *OSStorage* are base Smalltalk classes which you should *not* migrate. The remaining *OSObject* subclasses were defined by you or your team and must be migrated before you can use them.

Next, change the superclass for each class that requires migration:

1. From a browser, select the class to be migrated.
2. Change the class definition from

```
OSObject subclass: #MyVersion3Class
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
```

   to

```
OSPtr subclass: #MyVersion3Class
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
```

   If the class being migrated requires code page conversion support, use *AbtForeignOSObject* as a superclass as follows:

```
AbtForeignOSObject subclass: #MyVersion3Class
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
```

3. Save the new class definition.
4. Version the migrated class.

Migrating code that instantiates OSObject directly

If your Version 3.0 applications directly instantiate *OSObjects*, you must migrate the *OSObjects* used in this manner as follows:

1. From the System Transcript, select **Browse References** from the **Tools** menu.

2. In the displayed prompter, type `OSObject` and select **OK**. A References browser opens.
3. Scroll through the list of methods that reference *OSObject* and locate references in your own classes.
4. For each method requiring migration, change all occurrences of *OSObject* to *OSPtr* and save the method.
5. Version all classes changed as a result of completing step 4.

View wrappers that use attribute-to-attribute connections

When a view wrapper creates its view, it initializes the attribute-to-attribute connections in the parent part. In VisualAge Smalltalk V3.0, this initialization forces the view wrapper to always be the target of the alignment whether or not it is the target of the connection, and also whether or not the connection is unidirectional. In VisualAge Smalltalk V4.0, this initialization changed to make it consistent with the initialization of any attribute-to-attribute connection. This means that if the target attribute is read-only or the source attribute's value is *nil*, and the connection is bidirectional (source attribute is not read-only and the target attribute's value is not *nil*), the source of the connection is aligned with the target; otherwise, the target of the connection is aligned with the source.

This change may cause changes in the behavior of your Version 3.0 applications. To minimize the cases where there is behavior change between VisualAge Smalltalk V3.0 and VisualAge Smalltalk V4.5, the current VA Smalltalk migration will reverse all connections involving a view wrapper where the view wrapper was a source. At the end of migration, all connections involving view wrappers will have the view wrappers as targets.

The following table summarizes the different cases with results in VisualAge Smalltalk V3.0 compared to results in the current version of VA Smalltalk, and the last column **Need user attention** indicates the cases you might want to look at after migration.

Connection
Parent A > Wrapper A
A unidirectional connection from attribute A of the parent part to attribute A of the view wrapper

Source readOnly or target readOnly	Target readOnly or source nil	V3.0 result	VA Smalltalk result	Migration restores V3.0 result	Needs your attention
true	true	No alignment	No alignment		
true	false	Parent A > Wrapper A	Parent A > Wrapper A		
false	true	No alignment	No alignment		
false	false	Parent A > Wrapper A	Parent A > Wrapper A		

Connection
Wrapper A > Parent A
A unidirectional connection from attribute A of the view wrapper to attribute A of the parent part

Source readOnly or target readOnly	Target readOnly or source nil	V3.0 result	VA Smalltalk result	Migration restores V3.0 result	Needs your attention
true	true	No alignment	No alignment		
true	false	Parent A > Wrapper A	Parent A > Wrapper A		
false	true	No alignment	No alignment		
false	false	Parent A > Wrapper A	Parent A > Wrapper A		

Connection
Parent A <> Wrapper A
A bidirectional connection from attribute A of the parent part to attribute A of the view wrapper

Source readOnly or target readOnly	Target readOnly or source nil	V3.0 result	VA Smalltalk result	Migration restores V3.0 result	Needs your attention
true	true	No alignment	No alignment		
true	false	Parent A > Wrapper A	Parent A > Wrapper A		
false	true	No alignment	Wrapper A > Parent A The value of A in the parent part is set to the value of A in the view wrapper.	X	X
false	false	Parent A > Wrapper A	Parent A > Wrapper A		
true	true	No alignment	No alignment		
true	false	No alignment	Wrapper A > Parent A The value of A in the parent part is set to the value of A in the view wrapper	X	X
false	true	Parent A > Wrapper A	Parent A > Wrapper A	X	X
false	false	Parent A > Wrapper A	Wrapper A > Parent A The value of A in the parent part is set to the value of A in the view wrapper.		

Migrating the default font for Report Writer reports

For reasons of National Language Support, in VisualAge V4.0 the default font in the Report Writer was changed to a system font. If your Version 3.0 reports use the default font settings in the Report Writer, you must take one of the following actions to preserve your existing report layouts:

1. Change the default Report Writer font to its V3.0 default by doing the following:

   a. From the Organizer, select **Preferences** from the **Options** menu.

   b. In the VA Smalltalk Preferences window, select the **Report** tab.

   c. Select the **Change default font** button.

   d. In the Font Selection window, change the font to **display-times new roman-medium-13** and select **OK**.

   e. In the VA Smalltalk Preferences window, select **OK**.

2. Set the font in the Report Shell part for each report:

   a. In your report part, open the settings for the Report Shell part.

   b. In the Properties window, select *deviceFont* and then select the button that appears.

   c. In the Font Selection window, change the font to **display-times new roman-medium-13** and select **OK**.

d. Save the report part.

Packager warning 'No implementors of fixupNlsPool'

When you migrate an application to VA Smalltalk that has been around since either VisualAge Smalltalk V1.0 or V2.0, you may get a packager warning saying there are *'No implementors of fixupNlsPool'* when trying to package. If you are packaging as an IC, this will result in a runtime error when an attempt is made to run the packaged application.

To stop the error, browse implementers of *initializeAfterLoad* in your image. If there are implementers of this method, which is generated by VA Smalltalk, in any of your view classes you can delete them. This is obsolete code that was generated in previous VisualAge releases and is no longer needed. Deleting the implementers should clear up this particular packager warning and your application should run if packaged properly.

EwIconTree migration may cause walkback

Under certain circumstances, users who migrate from VisualAge Smalltalk V3.0 to VA Smalltalk may experience walkbacks associated with changes in EwIconTree. The walkbacks will look like the following:

```
Exception: (ExCLDTIndexOutOfRange) Index out of range.
```

There were many bug fixes as well as major changes made in VisualAge Smalltalk V4.0 for EwIconTree. Making these fixes caused some code that previously worked to fail with a walkback. The code should have failed all along, but due to bugs in VisualAge Smalltalk V3.0, the code actually worked fine.

The problem occurs if an application deletes items from the tree using a script such as the one that follows below:

In the following example, *self* is an instance of an *AbtAppBldrView*, *selectedItemFromTree* is the domain object, and *myContainerIconTree* is an instance an *AbtContainerIconTreeView.*

```
delete

self selectedItemFromTree notNil
  ifTrue: ["remove the object from the OrderedCollection that contains it"
    self selectedItemFromTree delete.
    "set the instance variable to nil"
    self selectedItemFromTree: nil.
    self myContainerIconTree deselectAllItems;
    refreshAllItems].
```

I found these methods to be interesting on EwIconTree.

```
deleteItem: item
"Delete an item from the list. This message is not supported in
tree lists. Rather, the application must delete the item from the
logical list itself, then call #deletedAt:count:."
```

```
^self error: (NlsCatEWe indexedMsg: 1) "$NLS$ Not valid for tree widgets"


deletedAt: position count: count
"Item(s) were deleted from the collection of root items at the
specified position. The position is specified as an index in the
collection of root items only. Refresh the list.

Usage:
This API can be used after the widget is created.

position
The index of the first item that was deleted.

count
The number of items deleted."

"Fully refresh the tree by resetting the items collection."
self items: self items.
```

The following comment from above does seem appropriate here:

Rather, the application must delete the item from the logical list itself, then call #deletedAt:count:."

So, if the items in the tree are refreshed by resetting the items collection, then the code will work.

The delete method should be changed to look like the following:

```
delete

self selectedItemFromTree notNil
ifTrue: ["remove the object from the OrderedCollection that contains it"
  self selectedItemFromTree delete.
  "set the instance variable to nil"
  self selectedItemFromTree: nil.
  self myContainerIconTree widget items:
   (self myContainerIconTree widget items).
  "self myContainerIconTree deselectAllItems;
  refreshAllItems"].
```

By making this change, you should be able to turn #refreshEntireListOnChange to false (the default) and it should work. Also notice that I commented out #deselectAllItems and #refreshAllItems; they are not needed. However, the code also works if those lines of code are left in.

In summary, due to bugs in V3.0 Ew containers, it is possible some application code will work in V3.0 when in fact it should have failed because it doesn't use the #deleteAt:count: method technique for deleting items. Furthermore, the changes in V4 that fixed these bugs will break this code. Thus, as a result of fixing problems with the EwIconTree, this problem is now apparent.

Solution

"Fully refresh the tree by resetting the items collection."

```
self items: self items.
```

Shortcut keys eliminated from Buttons

In Windows terminology (see Windows Desktop Applications Glossary) there are 2 types of accelerators:

1. Access keys (Alt+character) are used on Push Buttons, Toggle Buttons, and Radio Buttons, either on the canvas or in a menu.
2. Shortcut keys (Alt+character, Ctrl+character, Shift+character, or certain special keys) are used only on menus.

The character associated with an access key is also referred to as a 'mnemonic' and results in the mnemonic character being underlined in the label associated with the button (if underlining is enabled by Windows).

In menus, an entry can have both an access key and a shortcut key.

AbtPushButtonView and AbtToggleButtonView do not have properties specifying shortcut keys unless they are part of a menu (AbtCwMenuView.)

Migrating from Version 4.0, 4.01, or 4.02

This chapter addresses concerns for users of VisualAge Smalltalk Versions 4.0, 4.01, or 4.02 who are migrating to the current release of VA Smalltalk

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Packager changes
- Migrating distributed applications
- Other migration concerns

Packager changes

For Version 4.5, the packager acquired a new symbol and class adjustment API, a new priority for packaging instructions, and new problem and rule policies. If you are migrating from Versions 4.5, you can ignore this section.

Symbol and class reference adjustment API

Applications and subapplications can specify symbol and class reference adjustments method-by-method to effect a reduction algorithm by implementing *epSymbolReferenceAdjustmentsFor:in:* or *epClassReferenceAdjustmentsFor:in:*.

These methods allow applications to specify symbol and class references to ignore or add. In Version 4.02, the method *epClassReferenceAdjustmentsFor:in:* returned a list of classes. Since Version 4.5, *epClassReferenceAdjustmentsFor:in:* has had to return a collection of class *names*.

Priority for packaging rules

Packaging rules are sequenced and applied from the lowest priority to the highest priority. A rule's priority is determined by where it is defined. Rules defined by the packaging instruction method *packagingRulesFor:* have the highest priority. The priority of rules defined by applications is determined by the application prerequisite chain, with dependent applications having the higher priority. Within a method that defines rules (a rule store), the relative order of the rules determines priority, from lowest to highest.

The rule store methods that application might define have the following relative priority, shown from lowest to highest:

- *packagerIgnoreReferencesInSelectors*
- *packagerKnownSymbols*
- *packagerIgnoreSelectors*
- *packagerIncludeSelectors*
- *packagerIncludeClassNames*
- *packagerIncludeDoitMethods*
- *packagerIncludeMethods*
- *packagerRulesFor:*

If a higher priority rule conflicts with a lower priority rule, the higher priority rule takes precedence. For example, if *Kernel* specifies to exclude the class *Bag*, but *CommonWidgets* specifies to include the class *Bag*, the class *Bag* will be included.

When you select a component in the Image Contents Browser, any rules referring to that component are displayed. The rules are qualified as applied, overridden by higher priority rules, disallowed by the rule policy, or not applied due to an unsatisfied condition. If a rule becomes overridden, the higher priority rule that overrode it is shown.

Problem and rule policies

Packaging instructions are compatible with the prior release of the packager. However, some policies were replaced for Version 4.5. The packager detects the obsolete policies and replaces them with the appropriate policy.

Problem policies

Since Version 4.5, the following problem policies have been obsolete, and replaced with the "Configurable Problem Policy" (*EpConfigurableProblemPolicy*):

- EpImageComponentFilterSymbolArgumentsProblemPolicy
- EpImageComponentProblemPolicy
- EpRuntimeFilterSymbolArgumentsProblemPolicy
- EpRuntimeProblemPolicy
- EpNullProblemPolicy

Rule policies

Since Version 4.5, the following rule policies have been obsolete, and replaced with the "Explicitly Reduced Image Component Rule Policy" (*EpExplicitlyReducedImageComponentRulePolicy*):

- *EpExplicitXDImageComponentRulePolicy*
- *EpExplicitXDStartupImageComponentRulePolicy*

Migrating distributed applications

Since Version 4.5, the functionality provided by the VisualAge Smalltalk Distributed feature has been provided by the Server Smalltalk (SST) framework. This section helps you migrate applications made with the Distributed feature to the SST feature.

Where the Distributed and SST features provide a comparable level of functionality, they part dramatically in their implementations and APIs. This section lists the major functions and APIs used by Distributed feature applications and states the equivalent functions and APIs of SST.

To migrate your Distributed code from a Version 4.02 or earlier product, in some cases you will simply need to replace one API method with another. In other cases, you will need to make more dramatic changes to your application to accomplish the migration. The level of migration will vary significantly, depending on the application and the Distributed feature functions and API that are used. If an application merely references a few well-know global objects, the migration costs will be low. On the other hand, if the application uses some of the advanced Distributed Feature functions, such as the Distributed Load or the Parallel Processing API, then the migration will take more effort.

This section touches on the major functions of the Distributed feature, though it does not cover all of the functionality of SST. SST is only described where it overlaps with the Distributed feature. For example, SST allows many different configurations to control its behavior. For this section, only the SST configuration that matches the Distributed feature is covered. This configuration includes by-reference marshaling, full import or export of object spaces, and logical process dispatching. Other functions of SST that are not available in the Distributed feature, such as the Java RMI support, are not covered in this section. Further, the descriptions provided here are for the purposes of understanding how to migrate your code. The descriptions are not meant to fully cover the Distributed feature or SST. In other words, consult the appropriate VA Smalltalk guide for more information.

This section covers the major functions of the Distributed feature in three sections:

- Distributed Smalltalk infrastructure
- Distributed development tools
- Distributed runtime APIs

Distributed Smalltalk infrastructure

The Distributed feature provides a base infrastructure for distributed object applications. This infrastructure provides remote object messaging between different Smalltalk object spaces.

The following components of the Distributed feature make up the infrastructure layer. For each component, a brief description and the equivalent SST component are given.

Object space

Description

An object space is a collection of active Smalltalk objects loaded from a single Smalltalk image file. The Distributed feature object spaces are represented by the classes *DsLocalObjectSpace* and *DsRemoteObjectSpace*.

In the Distributed feature, the object space and its location are defined through the name server. A key and TCP/IP host name or address identifies an object space.

SST equivalent

SST also uses the concept of object spaces. In fact, it provides two types of spaces: simple object spaces and full object spaces. The equivalent of the Distributed feature object space is the full object space, which maintains both an import and export set. The SST classes *SstLocalSpace* and *SstRemoteSpace* are used for full object spaces.

In SST, the management of object spaces is handled through applications contexts. Contexts maintain the distributed environment for an SST application and provide isolation from other applications. Object space locations are managed through machine objects and URL strings. URL strings take the format `scheme:/transport/address`. An example URL string is `myScheme:/tcp/myserver.ibm.com:2345`.

Communications

Description

The communications between object spaces in the Distributed feature is performed with the TCP/IP communications protocol.

SST Equivalent

SST provides an open communications component that is transport independent. Currently supported transports include TCP/IP, HTTP, and the e-mail transports SMTP and IMAP4.

Marshaling

Description

As part of the remote object messaging, objects passed by-value are converted to and from byte representations. The Distributed feature marshaling is based on the Swapper tool, which is described in the *Smalltalk User Guide*.

SST equivalent

SST has two general purpose message marshalers. The *SstSwapperMarshaling* is based on the Swapper. The *SstLightweightMarshaling* is optimized for the smaller messages typical in a distributed object application.

Logical process

Description

The Distributed feature uses logical processes to maintain synchronous message sending across different object spaces. Logical processes maintain the Smalltalk process model across multiple object spaces.

SST equivalent

SST can be configured with several different policies for message dispatching. The *SstLPDispatcher* can be used to support logical processes.

Activation

Description

The Activator is a background process, or daemon, that must be running on a system that can receive incoming requests from other Distributed systems. The activator is a stand-alone executable program, separate from the Smalltalk image. The activator has two main purposes:

- To route incoming requests to the correct target object space on its local machine
- To start the target object space, if necessary

SST equivalent

As an alternative to the Activator program, SST server object spaces allow for connections directly from client object spaces. This feature simplifies the setup required on the server and speeds up the connection time. Without a front-end program like the Activator, SST does not support automatic start up of the server object space, triggered by a client request. Also, to run multiple object spaces on the same server machine, SST requires that each server object space be defined with a different URL. For applications using the TCP/IP transport, this would mean that each server object space would require a different port number and the clients would need to know the port number of the server object space they desire.

Security

Description

The Distributed feature provides several levels of security protection for distributed applications. Three levels of security are built into the distributed feature: basic authorization, password-based client authentication, and mutual password-based authentication. In addition, on OS/2 only, external security is provided through the Generic Security Services (GSS) API.

SST equivalent

At this time, SST does not provide any built-in security protection. You can implement your own security through object space connection callbacks and through extensions to the marshaling framework.

Distributed development tools

The tools and functions described below are provided as part of the Distributed feature development environment. For each tool or function, a brief description and the equivalent SST tool or function is given.

Remote Workspace

Description

A Remote Workspace is an extension of a standard Smalltalk workspace window. It provides a local window with the context of a remote object space. Smalltalk code executed in a remote workspace window compiles and runs in the target remote object space. Results are shown in the local window.

SST equivalent

The remote workspace tool is not currently available in SST.

Remote Transcript

Description

A Remote Transcript has the same functionality as the standard Smalltalk Transcript window, except that it executes within the context of a remote object space. Similar to the Remote Workspace, Smalltalk code executed within a Remote Transcript window runs in the object space associated with the Remote Transcript. In addition, the Remote Transcript window echoes all messages displayed to the remote object space's real Transcript.

SST equivalent

A Remote Transcript is not available in SST.

Remote file dialog

Description

A remote file dialog is a file dialog that enables you to access files that reside on remote machines. The remote file dialog is accessed through the **Open** choice of a **File** menu on the Remote Workspace or Remote Transcript windows. For example, you can use a remote file dialog to open a remote text file in a Remote Workspace, make changes, and then save the file on the remote machine.

SST equivalent

The Remote File Dialog function is not available in SST.

Distributed Inspector

Description

The Distributed Inspector is an extension to the base Smalltalk inspector. It allows remote objects to be inspected through a local inspector window. As with local objects, remote object can be browsed and modified. The distributed inspector is provided through the *DsInspector* class.

SST equivalent

SST provides a comparable distributed inspector through the *SstInspector* class.

Distributed Debugger

Description

The distributed debugger is an extension to the base Smalltalk debugger. It allows logical processes to be debugged across object space boundaries. The full function of the base Smalltalk debugger is available in the distributed version, including remote breakpoints, remote step functions, and inspecting remote objects. The distributed debugger is provided through the *DsDebugger* class.

SST equivalent

SST provides a full function distributed debugger through the *SstDebugger* class.

Distributed garbage collection

Description

The distributed garbage collection function is used to remove distributed objects that are no longer referenced by any object space. Distributed garbage collection performs the same role as the base Smalltalk garbage collection function, with the exception that cycles are not collected. For example; object x, in object space A, references object y, in object space B, and vice versa. Even with no other references, since the objects form a distributed cycle, they will not be collected. Users are required to break such cycles.

SST equivalent

SST provides distributed garbage collection equivalent to the function provided in the Distributed feature. There are, however, differences in the setup procedures. The Distributed feature distributed garbage collector is started automatically in each object space. In SST, one object space needs to be specified as

the coordinator space for the distributed garbage collector. Additionally, each object space must start and shutdown the distributed garbage collector with code similar to the following:

```
(SstDgcConfiguration new
   instantiateIn: context coordinatorSpace: coordinatorSpace) startUp
(SstDgc for: context) shutDown
```

Distributed load

Description

The distributed load tool enables you to load, from one machine, applications into different object spaces. Distributed load is based on a distribution matrix, which is an extension to the Smalltalk configuration maps. A distribution matrix enables you to specify object space locations for each application in the configuration map. The object space location determines which object spaces, in the network, the specified application will be loaded into. For each object space location, each application in the configuration map has a designation for a full load, no load, or a proxy load. A proxy load specifies that the application is not fully loaded in the location, but instead, the globals of the application are loaded as shadows (or proxies), pointing to where the application is actually loaded. The distributed load function makes it easy to develop applications locally and then, through the distributed matrix, load them into other object spaces, without changing any code.

SST equivalent

SST participates in the Cross Development (XD) environment. XD can be used to load and package applications targeted for another environment. The user then moves the XD packaged image to the target machine to run it. The additional function that you get with distributed load, which you do not get with XD, is the transparency provided by the proxy load of an application. SST applications must explicitly know and reference those objects that are remote.

Name server

Description

A distributed name server is a specialized dictionary object that you can use to store, manage, and retrieve object references based upon symbolic names. The symbolic names provide an indirect way of referring to object references without having to specify physical location information. A name server entry consists of an identifier and a reference type, as follows:

Global
Refers to a global object, including classes

DictionaryItem
Refers to an item in a dictionary

Location
Refers to an object space location

NameServer
Refers to another name server

By default, all name servers are temporary. Changes you make to the contents of a temporary name server are made only within the context of the current Smalltalk session. However, you can specify that any name server be persistent. A persistent name server stores its contents in a separate archive file outside the Smalltalk environment. Each time you start an object space containing a persistent name server, the name server reloads its contents from the archive file. Therefore, any changes to the contents of a persistent name server remain in effect even after shutting down and restart the Smalltalk image.

SST equivalent

SST provides a basic name server capability through its object export functions. However, there are no tools or user interface for managing exported objects. SST does provide a runtime API for managing exported objects. For more information, see Name server.

Event profiling (method calls)

Description

The Event Profiler is a text-based browser used to analyze application performance. It shows all of the events (method calls) sent among selected objects. You can use the Event Profiler to browse each event and trace the method call stack that led up to it. You can also use the Event Profiler to measure the number of messages that cross object space boundaries. The tool is very useful for reducing the number of remote message sends and determining the optimal boundaries to split an application between object spaces.

SST equivalent

SST invocation handlers have various callbacks for sending or receiving requests and replies. The following callbacks are available for adding additional processing:

- SstSendRequestCallback
- SstDispatchRequestCallback
- SstSendReplyCallback
- SstDispatchReplyCallback
- SstFirstIncomingCallback
- SstFirstOutgoingCallback

VA Smalltalk Parts

Description

The Distributed feature provides a sample distributed name server part to enable you to build a name server entry in your user interface and connect it to local or remote objects. You may use this part to refer to these objects.

SST equivalent

SST does not provide any VA Smalltalk Parts for remote objects.

Distribution menu

Description

The Distributed feature adds a **Distribution** menu to the System Transcript. This menu provides user interface controls for the operation of the Distributed environment. For example, there are menu selections for enabling or disabling the distributed debugger functions; resetting distributed processing; launching browsers for the name server and the event profiler.

SST equivalent

User interface controls for the SST operation are available from the **SST** menu choice found in the **Tools** menu of the System Transcript. The menu choices enable you to clear the distributed system, shut down the distributed system, and enable or disable the distributed debugger.

Distributed runtime APIs

The Distributed feature provides a runtime API for applications. The APIs are subdivided into categories. This section describes the API categories and, for each category, states the equivalent SST runtime APIs. Where needed, this section also provides information to assist you in porting from the Distributed feature runtime APIs to the SST runtime APIs.

Basic remote object API

Description

The Distributed feature provides a basic API for accessing remote objects. Some of the commonly used methods include--

globalAt:aKey
Answers the current value of the global value named by *aKey* in the object space represented by the receiver.

- *globalAt:* in *DsObjectSpace*
- *globalAt:ifAbsent:* in *DsObjectSpace*

dsInSpace:anObjectSpace
Answers an appropriate representation of the receiver in the specified object space. This representation may be the receiver itself, or a proxy (*DsRemoteObjectPointer*) to a remote object. If necessary, a copy of the receiver is made.

- *dsInSpace:* in *Object*
- *dsInSpace:* in *Class*

SST equivalent

Client object spaces can access remote global objects directly, without the need to have them exported on the server object space. The following API is defined to access remote globals.

sstGlobalIn:space
Answers a reference to the remote global object in @*space* whose global name is the receiver.

- *sstGlobalIn:* in *String*
- *sstGlobalIn:context:* in *String*

Example code to access a remote *Array* class in the object space *remoteOS* is as follows: `#Array sstGlobalIn: remoteOS`

Porting guidelines

Migration of the API for accessing remote global objects is straightforward. You can convert the Distributed feature *globalAt:* methods directly to the SST *globalIn:* methods.

Converting the Distributed feature *dsInSpace:* API depends on its use. If you use it for classes, you can use the SST *globalIn:* API since classes are also global variables. If you use the *dsInSpace:* API to copy local objects to the remote object space, SST does not have an equivalent API. You may be able to achieve similar results by using the SST marshalling wrapper *sstAsDeepValue*. The wrapper causes the local object to be copied when passed as a parameter to the remote object space.

Object copying API

Description

The Distributed feature provides several APIs you can use to create local copies of remote objects. By using these methods, you request that an object be passed by value rather than by reference, if it is possible. These methods include--

asDsByValueObject
By sending the message *asDsByValueObject* to an object, you request that the object be passed by value rather than by reference, if possible. For local objects, which would otherwise pass by reference, the *asDsByValueObject* method returns a proxy for the receiver. The proxy imitates the receiver object and causes the receiver to be passed by value, when sent to another object space.

asDsByValueArray
This message returns the receiver converted to an *Array* of class *DsByValueArray*. A *DsByValueArray* is always passed by value. The elements of the array are not affected and will be passed by their own default rules. Although the objects in the array might reside in other object spaces, the *DsByValueArray* itself is always local.

dsPerformWithLocalResult:
This message functions much like the base Smalltalk method *perform:*, except that the object returned is a local object. If the result of the *perform* is a remote object, it is copied to the local object space, if it is possible to do so.

SST equivalent

SST provides marshaling wrappers that specify how an object is to be passed to another object space. The following APIs will return a corresponding marshaling wrapper for the receiver object:

sstAsDeepValue
This method answers a deep value marshaling wrapper, on the receiver, containing the receiver and dictating that the receiver should be marshaled by deep copy.

sstAsShallowValue
This method answers a shallow value marshaling wrapper, on the receiver, containing the receiver and dictating that the receiver should be marshaled to one level of value.

sstAsReference
This method will answer a reference marshaling wrapper, on the receiver, containing the receiver and dictating that the receiver should be marshaled with a reference to the local object.

Porting guidelines

The Distributed feature object copying methods can be converted to SST as follows:

asDsByValueObject
Replace the *asDsByValueObject* method with the SST method *sstAsDeepValue*.

asDsByValueArray
Replace the *asDsByValueArray* method with the SST method *sstAsShallowValue*.

dsPerformWithLocalResult:
There is no equivalent function to *dsPerformWithLocalResult:*. Users of SST can achieve the same function as the *dsPerformWithLocalResult:* method by optimizing their application code. In the method executed on the remote object space, the result object can be returned by value, by sending it the *sstAsDeepValue* method.

In SST, the *sstAsDeepValue* and *sstAsShallowValue* methods return a wrapper object that does not imitate the behavior of the receiver. The wrapper should only be used as part of a remote message send. In contrast, for the Distributed feature the *asDsByValueObject* method returns a proxy object and the *asDsByValueArray* method returns an object of type *Array*. These objects will behave like the receiver, prior to their use in a remote message send.

Parallel processing API

Description

The Distributed feature provides a set of extensions to the base Smalltalk classes that facilitate performance optimization through parallel processing. These extensions consist of the following methods:

- *Collection*
  - *dtParallelDo:*
  - *dtForkEachAndJoin*
  - *dtForkEachAndContinue*

- *Number*
  - *to:dtParallelDo:*
  - *to:by:dtParallelDo:*

SST equivalent

SST does not supply the equivalent of the Parallel processing API.

Porting guidelines

Since these methods are implemented on top of the basic Smalltalk process model, you can achieve similar results by using the Process model API directly (*fork* in *Process*, *wait* in *Semaphore*, and so on).

Name server

Description

As stated in Name server, the Distributed feature provides a built-in name server. At runtime, several APIs are available to access object references from the name server. A partial list of the methods includes--

- *DtNameServerListDictionary*
  - *default*
  - *at:* key *put:* aValue
- *DtNameServerDictionary*
  - *at:* key
  - *at:* key *put*: aValue
  - *referenceAtFirstIdentifier:* anIdentifier
  - *keys*

The keys in the name server identify object references. These object references are used to create shadows (or proxies) for the remote object that they describe. Object references can also be represented with string values. The following methods are examples of the object reference API, implemented on class *DsGlobalReference*:

- *shadow*
- *repString*
- *repStringOn:* aStream
- *objectRefString:* aString
- *objectRefStringOn:* aStream

SST equivalent

In order to access well-known remote objects, SST provides primitive naming services through symbolic references. In server object spaces, objects can be exported through symbolic names. Clients can then access these remote objects by using their symbolic name and object space. For global objects, the server object space does not need to explicitly export the object. Remote global objects can be accessed with the following API implemented in *String*:

- *sstGlobalIn:* aSpace
- *sstGlobalIn:* aSpace *context:* aContext

For non-global objects, the server object space can export objects through a symbolic name. Client object spaces can access the remote object using that symbolic name. The following methods are part of the API, provided by object spaces, to manage exported objects:

- *export:* object
- *export:* object *as:* objectId
- *unexportId:* key
- *unexportObject:* object

Example code to make the server object space's date externally known, with the ID *#serverDate* includes--

```
serverObjectSpace export: Date today as: #serverDate
```

The class *SstRemoteReference* provides the following class method API to access remote objects by symbolic references:

- *export:* object *in:* space
- *for*: object *in:* space

The following example client code returns a remote reference to the object identified by *#serverDate* in *serverObjectSpace*:

```
SstRemoteReference for: #serverDate in: serverObjectSpace
```

Porting guidelines

The name server entries provided in the Distributed feature will have to be converted to symbolic references in SST. Global objects can be accessed directly, through the *globalIn:* API methods. Other objects will need to be explicitly exported by the server object space.

On server object spaces, setup code will be required to define the exported objects. On the client side, references made through the name server will need to be converted to SST API calls to access the global or exported objects.

Event callbacks (connect/disconnect)

Description

The Distributed feature provides a set of callbacks that you can use to keep track of the local object space's connections to other object spaces. These callbacks include the following:

- *dsObjectSpaceConnectCallback*
- *dsObjectSpaceDisconnectCallback*
- *dsObjectSpaceDisconnectOnExitingCallback*
- *dsObjectSpaceDisconnectOnStartCallback*

SST equivalent

SST provides callbacks from several of its components. The transport component, when used with connection oriented transports, provides the following callbacks for connect and disconnect:

- *SstConnectCallback*
- *SstDisconnectCallback*

Porting guidelines

For SST connective transports, such as TCP, the Distributed feature callbacks can be approximated with the *SstConnectCallback* and *SstDisconnectCallback* callbacks from the SST transport component.

You need to be aware that the SST callbacks are not an exact replacement for the Distributed feature callbacks. The Distributed feature callbacks are at the object space level and use object spaces as their *callData*. On the other hand, the SST callbacks are at the transport level and use endpoints as their *callData*.

Further, the Distributed feature identifies three distinct disconnect callbacks, based on when an object space disconnect occurs. If SST users need the same distinction, they will need to determine the reason for the disconnect as part of their callback processing.

Below is example code showing how to register for a *SstDisconnectCallback*:

```
transport configuration callbacks
   addCallback: SstDisconnectCallback
   receiver: myConnectionManager
   selector: #processDisconnectCallback:clientData:callData:
   clientData: myClientData
```

Distributed exceptions

Description

The Distributed feature defines a general exception, *unavailableObjectException*, which is signaled whenever an attempt to reach a remote object fails.

When exceptions occur on a distributed process and are not handled by the user, the default action is to report the error in the User Interface (UI) object space. The UI object space is defined as the first object space in the distributed logical process stack that is defined to have user interface capability. In a typical client/server application, this will be the client object space. For exceptions that occur in the server object space, the exception is propagated to the client object space, where the default error handler is invoked. For development time, this would be a distributed debugger. At run time, this would be an error pop-up.

SST equivalent

SST provides several exceptions for errors occurring during SST processing. The exceptions and their hierarchy are shown below. When a SST exception is raised, it will contain a description and objects related to the error.

(ExAll)

ExSstFatalError

(ExError)

ExSstInvocationError

ExSstNonFatalError

ExSstSetupError

ExSstUnknownException

Similar to the Distributed feature, exceptions on distributed logical processes are handled by a specified object space. In SST, the *SstErrorReporter* specifies the appropriate UI Handler to process the error. In a typical client-server application, this will be the client object space.

Porting guidelines

In the Distributed feature, the single *unavailableObjectException* can be used to catch all general distributed errors. In SST, you have the flexibility to catch exceptions in more specific categories. With SST, you can also catch very general distributed errors with *ExAll* or *ExError*.

Object space management

Description

The Distributed feature is built around the concept of object spaces. Local object spaces are used to manage the local image, of the distributed system. Likewise, remote object spaces are used to represent a remote image, of the distributed system. For the user, the object space, identified by an object space key, provides the means of location for objects. Through the use of the name server, the location of an object can be defined. To change the location of an object, only the entry in the name server needs to change. In addition, object spaces provides the base level of control for connection management. Object spaces are used to connect and disconnect images in the distributed system.

The Distributed feature provides methods to query object spaces and manage connections. The following API methods are available:

- *DsDistributedSystemConfiguration*
  - *reset*
  - *resetAllConnections*
  - *terminateAllConnections*
- *DsLocalObjectSpace*
  - *allObjectSpaceNames*
  - *allObjectSpaces*
- *DsRemoteObjectSpace*
  - *connected*

SST equivalent

SST also provides object spaces to manage remote objects. The SST object spaces manage the namespace and manage the exporting of objects to other object spaces. On top of the remote object model, SST provides application contexts to manage the distributed system on an application basis. Each application in an image can have its distributed environment separately managed through its own context. The APIs to create an object space requires a machine name and a URL to be supplied. The following partial list of APIs, on class *SstApplicationContext*, can be used to create, setup, and delete spaces:

- *addSpace:* spaceName *on:* machineName *at:* urlList
- *setupFor:* spaceName *using:* config
- *removeSpace:* oldSpace
- *spaceFor:* id
- *space*
- *spaces*

A URL string defines the element of location for object spaces. The URL string is used to set up an object space and defines information such as, invocation scheme, transport name, and transport-specific address. The general form is as follows:

```
scheme:/transport/address
```

A specific URL example for TCP is--

```
myByRefScheme:/tcp/foo.com:4000
```

The *SstApplicationContext* handles the management of the distributed system. The context can be used to start up and shutdown the distributed system through the following API methods on class *SstApplicationContext*:

- *startUp*
- *shutDown*
- *clear*

Porting guidelines

The Distributed feature uses the *DsDistributedSystemConfiguration* to manage the distributed environment. The application contexts, in SST, provide the same level of controls, but allow for multiple contexts in a single distributed system. For example, the `DistributedSystem reset` message can be replaced with the SST message `myContext shutDown; startUp`.

The specifications of object space locations are handled differently. The Distributed feature defines the object space locations in its name server. The object spaces are accessed by a name server key. When the location of an object space changes, only the name server needs to be updated. A persistent name server file can be used for application deployment on different machines. On the other hand, SST requires that the object spaces be created with their machine and location information supplied in a URL string. If the location of an object space changes, a new URL string will need to be supplied to SST. It is the applications responsibility to provide the correct URL string to SST. To support application deployment on different machines, SST applications will need to acquire the correct URL strings, at run time. Some examples for specifying URL strings are through command line parameters, entries in an ini file, or external name servers.

Distributed runtime APIs

The following methods are the officially supported APIs of the Distributed feature. The API methods are in the categories **DS-API** and **DT-API**.

DS-API

- *Object*
  - *dsInSpace:*
  - *asDsByValueArray*
  - *asDsByValueParameter*
- *Class*
  - *dsInSpace:*
- *DsLocalObjectSpace*
  - *globalAt:* ifAbsent:
  - *addCallback:receiver:selector:clientData:*
  - *removeCallback:receiver:selector:clientData:*
  - *allObjectSpaceNames*
  - *allObjectSpaces*
- *DsObjectSpace*
  - *globalAt:ifAbsent:*
- *Collection*
  - *asDsByValueArray*
  - *asDsParameterArray*
- *EsString*
  - *asDsByValueParameter*
  - *asDsByValueObject*
- *DsByValueParameter*
  - *asDsByValueParameter*
  - *asDsByValueObject*
- *DsCallbackRec* class
  - *dsObjectSpaceConnectCallback*
  - *dsObjectSpaceConnectionFailedCallback*
  - *dsObjectSpaceDisconnectCallback*
  - *dsObjectSpaceDisconnectOnExitingCallback*
  - *dsObjectSpaceDisconnectOnStartupCallback*
- *DsDistributedSystemConfiguration*
  - *reset*
  - *resetAllConnections*
  - *terminateAllConnections*
  - *unavailableObjectException*
- *DsRemoteObjectSpace*
  - *connected*
- *DsObjectSpaceDescriptor*
  - *repString*
  - *repStringOn:*
- *DsGlobalReference*
  - *copyWithKey:*
  - *objectRefString:*
  - *objectRefStringOn:*
  - *remoteObjectIn:*
- *DsObjectReference* class
  - *fromRepString:*

- *DsObjectReference*
  - *objectSpaceDescriptor*
  - *repString*
  - *repStringOn:*
- *ProcessorScheduler*
  - *dsActiveLogicalProcess*
  - *dsActiveUiObjectSpace*

DT-API

- *DsGlobalReference*
  - *objectSpace*
  - *objectSpaceDescriptor*
  - *shadow*
- *DtDictionaryItemReference*
  - *objectSpaceDescriptor:*
- *DtNameServerDictionary*
  - *at:*
  - *at:* aKey *ifAbsent:* aBlock
  - *at:put:*
  - *keys*
  - *localAt:*
  - *localAt:ifAbsent:*
  - *localKeys*
  - *nextNameServer*
  - *nextNameServerKey*
  - *nextNameServerKey:*
- *DtNameServerListDictionary*
  - *at:put:*
  - *default*

Other migration concerns

This section describes how to handle other problems you might encounter when migrating your applications.

Using VA Smalltalk features in server applications

If your server applications will use the database, communications, C or COBOL language, or other VA Smalltalk feature, ensure that you load the VA Smalltalk feature and not just the VA Smalltalk Base feature. Thus, to use the Database DB2 CLI support, load the **VA: Database, DB2 CLI** feature instead of the **IBM ST: Database, DB2 CLI** feature. This applies even if you are coding in Smalltalk and not using nonvisual parts.

Runtime code for nonvisual parts in server applications

The runtime format of public interfaces for nonvisual parts has changed. The changes affect the *#IS_...* methods generated for parts.

If you have nonvisual parts that you developed in a Version 4.02 or earlier application and you want to use those parts in a server application, you must re-generate the parts' runtime code. To do this, select the parts in the Organizer and then select **Generate > Runtime Code** from the **Parts** menu.

Headless runtime applications

Support for creating headless runtime applications has been provided by the Server Workbench feature since Version 4.5. This capability is no longer part of the base product. Application EpHeadlessRuntimeStartUp has been removed. The equivalent function is provided by AbtHeadlessRuntimeStartUp.

Signalling events with arguments

In VA Smalltalk it is possible to signal an event with arguments using a method such as

```
self signalEvent: anEventName with: aValue
```

Objects are able to register themselves as dependents of an object and be notified when an event is signalled. One use of this is in the Composition Editor where an attribute-to-attribute connection is made. When either the source or the target signals the event associated with the attribute the other end of the connection will perform its set method associated with the attribute. For example, you can connect the *object* attribute of a text box to the *object* attribute of a label and when the text box signals its *object* event the label will update its object.

Prior to Version 4.5, the argument value that was passed into the *with:* keyword was ignored by the attribute connection which always went to the object that was raising the event and got the current value of the attribute. This had the effect that in some circumstantes the get method associated with the attribute would be triggered multiple times and also that you had no way of signalling an attribute-to-attribute connection to align itself with anything other than the current attribute value. Since Version 4.5, the value that is used as the argument to the *with:* keyword has been used to refresh the attribute, avoiding the need for the *get* method to be performed on the object that is raising the event.

You should be aware of this change in behavior as it will now expose methods that were previously passing incorrect values into the *with:* argument. For example, consider the method

```
firstName: aString
   firstName := aString.
   self signalEvent: #firstName with: aString.
```

This is the correct syntax. The event *firstName* is being signalled with the current value. Next, consider

```
firstName: aString
   firstName := aString.
   self signalEvent: #firstName with: self
```

This is incorrect. The event is signalled with *self* as the argument. If any methods exist that pass incorrect values to the *with:* keywords, they should be changed to either pass the correct arguments, or the keyword dropped.

```
firstName: aString
   firstName := aString.
   self signalEvent: #firstName
```

If there are no *with:* arguments, then the attribute-to-attribute connection will get the current value of the *firstName* attribute (by performing the get method) to align itself. It is preferable, however, to pass the value to the *with:* keyword if it is readily available.

To look for potential problems, evaluating the following methods will open browsers that show all methods that are signalling events with arguments. These can be inspected to make sure that the argument value used on the *with:* keyword is the correct one.

```
System allMethodsSending: #signalEvent:with:.
System allMethodsSending: #signalEvent:with:with:.
System allMethodsSending: #signalEvent:withArguments:.
```

Ending server applications

In the server environment, an application does not automatically end when the UI process terminates. The application will not terminate so long as there are any active processes. It is recommended that you explicitly end your application by doing one of the following:

Batch
Code `System exit` or `System exit:withObject:` at the point where your application is complete.

CICS
Code `CICS return exec` at the point where your application is complete.

IMS
Code `System exit` or `System exit:withObject:` at the point where your application is complete.

MVS applications that reference subsystem type 'TM'

In Versions 4.0, 4.01, and 4.02, the XD image types included--

- CICS MVS Target
- CICS OS/2 Target
- CICS Windows NT Target
- Native MVS
- Native MVS (OS/2 Target)
- Native MVS (Windows NT Target)
- IMS MVS Target

In Version 4.5, the image types changed to the following:

- MVS
- MVS Simulation (OS/2 Target)
- MVS Simulation (Windows NT Target)
- OS/2
- Windows NT
- AIX

The image types now apply to the platform they represent. The CICS and IMS API's must be loaded into the image type as a feature. If your application checks the value of the subsystem type "TM" when loaded into an image, do either of the following before loading your application into a Version 6.0 image: load the CICS or IMS feature or change your application so it checks for a different subsystem type at runtime.

Smalltalk error message in a CICS workstation environment

When running a Smalltalk application in CICS for OS/2 or CICS for Windows NT, an error message is written to the console. Since Version 4.5, these error messages have been written to the CSMT Transient Data Queue.

Search path for multiple images of Smalltalk using CICS classes

Images created with pervious releases of Smalltalk will continue to run in Version 6.0, but images created with Version 6.0 will not run in a previous release of Smalltalk. Thus, if you have more than one release of Smalltalk on your workstation, you must place the directory for Version 6.0 first in your workstation's search path.

Snapshot files

Snapshot files list the contents of an image. You typically use snapshot files for packaging image components (ICs) or for just keeping a record of what applications, classes, and methods are in an image at a particular time.

Snapshots created with a Version 4.01 or 4.02 image are not compatible with the Version 6.0 packager. Thus, you cannot package with prerequisite ICs created with the Version 4.01 or 4.02 packager. You must repackage all ICs, and thus shapshots, using a Version 6.0 image.

Migrating from Version 4.5

This chapter addresses concerns for users of VisualAge Smalltalk Version 4.5 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Formatted Text Field part
- Enhanced shared library support on AIX
- Server applications using .mpr files

Note:

Beginning with VisualAge Smalltalk V5.0, ODBC Drivers are no longer shipped with the product. You can check your Database Resource Manager for ODBC drivers. Most, if not all, major DBRMs now ship with ODBC drivers.

Formatted Text Field part

The Formatted Text Field part's behavior for Monetary format has been changed to bring it into line with customer expectations. Prior to VisualAge V5.0, the object value of this part was held in either integer or floating point format depending on whether the entered value contained a decimal point. This lead to possible inaccuracies in the results, particularly in applications which combined multiple object values (for example, summing values which included both very large values and very small values containing decimal points).

In the past, we suggested that you convert the object value to *ScaledDecimal* before using it. This workaround solved most, but not all, problems associated with holding the object value as a floating point number.

The behavior of the Formatted Text Field part is now different when its format is Monetary. In this situation, its object value is always held in and exposed as a *ScaledDecimal* number. The effect of this change on your applications depends on the application.

If your application was developed following our recommendation of converting the object value to *ScaledDecimal* (using *asScaledDecimal*), you do not need to make any changes since the *asScaledDecimal* instance method in *ScaledDecimal* answers *self*. It is possible that, for certain obscure values, your application will receive a number that is either larger or smaller by 1 in the least significant position. The value answered in VA Smalltalk is correct while the value answered previously was incorrect.

If your application was developed using the object value of the Formatted Text Field (Monetary format) as answered by the part, you should examine the application logic to determine the effect of this change.

As part of this change, the obsolete *AbtIntegerFormat* and *AbtFloatFormat* classes have been moved to the *VisualAge, Obsolete Parts* configuration map. These classes formed the basis of the VisualAge Smalltalk V3.0 Formatted Text Field (Monetary format) support. If you need these classes in your application, you will need to load the *VisualAge, Setting Views* and *VisualAge, Obsolete Parts* configuration maps into your development image.

Enhanced shared library support on AIX

In older versions of the AIX operating system, shared libraries were not supported. A work-around solution was to include an *EsUserPrimitive* table and make this table the entry point for the library module. The `.a` suffix was used by default for libraries of this type. These special load modules have been commonly referred to as "willie" files and given an explicit `.w` suffix to differentiate them from normal load modules with the `.a` suffix.

The currently supported versions of AIX have implemented full shared library support. Therefore, VA Smalltalk searches first for a library with a `.so` prefix, and if it finds a file of this name it will use the operating system's dynamic load mechanism to access the specified shared library. If it fails to find a `.so file`, the search continues looking for an `.a` library and assumes that an `.a` file implements the work-around solution.

The current search algorithm for loading shared objects and load modules is as follows:

1. Call dlopen with the name specified in the *PlatformFunction*.
2. On fail, add the `.so` suffix to the name specified in the *PlatformFunction* and call dlopen.
3. On fail, add the `lib` prefix and the `.so` suffix to the name specified in the *PlatformFunction* and call dlopen.
4. On fail, call load with the name specified in the *PlatformFunction*.
5. On fail, add the *.a* suffix to the name specified in the *PlatformFunction* and call load.

In the past, we recommended that you always let the virtual machine supply the suffix as this allows you to write Smalltalk code that is not bound to a particular operating system. However, with this new shared object support, it is critical that you follow this recommendation if you are attempting to access a "willie" file. This is because the AIX dlopen function successfully opens a "willie" file in Step 1 above. However, it does not successfully find the correct entry point. There is no way to force a "willie" file with an explicit suffix to resolve at Step 4.

If you use "willie" files on AIX and if you code an explicit suffix on the *libraryname* parameter of the *PlatformFunction* creation method, you must change this method call by removing the explicit suffix. This may also entail renaming your "willie" file.

Server applications using .mpr files

VA Smalltalk server applications no longer need message and pool repository (`.mpr`) files unless the strings in the applications are translated.

If your server application no longer needs `.mpr` files, you should regenerate *abtInternals* in the server application. (From the **Parts** menu of the Organizer, select **Generate > Runtime code**.)

Migrating from Version 5.0

This chapter addresses concerns for users of VisualAge Smalltalk V5.0 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Some base features no longer available
- Some separately priced features no longer available
- Hiding Java packages and projects in browsers
- Providing a customized splash screen
- Migrating feature control files
- Importing feature code
- Migrating method mappings to JDK 1.2 or later
- Changes in Server Workbench
- Current code page support is inconsistent on supported platforms
- AbtConnectionSpec>>#targetCodePage: now expects a String as an argument

Some base features no longer available

The following Base and Server Workbench features are no longer available:

- Visualization Tools
- Multimedia Support
- Tivoli Connection
- NetBIOS
- RPC
- Notebook Style Settings View
- VisualAge Obsolete Parts
- Universal Lightweight Client (ULC)
- CICS Simulation Workstation

Some separately priced features no longer available

The following separately priced features are no longer available:

- VisualAge Smalltalk ObjectExtender
- VisualAge Smalltalk UML Designer
- VisualAge Smalltalk Advanced Database Feature

Hiding Java packages and projects in browsers

Beginning with VisualAge Smalltalk V6.0 the following browsers, and dialogs associated with them, will not display Java packages or projects:

- Configuration Maps Browser
- Application Editions Browser
- Application Manager
- Organizer

As customers begin to share the library manager between VisualAge Smalltalk and VisualAge for Java, it would be difficult to use the Smalltalk browsers if Java Projects and Packages appear in the Smalltalk browsers. Further, because Java Projects and Packages are not Smalltalk code elements, they cannot be reliably managed using the Smalltalk browsers. Therefore, we have hidden these Java code elements from displaying in the Smalltalk browsers listed above. You should use VisualAge for Java to manage your Java Projects and Packages.

Providing a customized splash screen

Starting in VisualAge 6.0 the Windows splash bitmap is read from a file rather than from a resource linked into the executable. This bitmap file must have the same filename as the executable, must be a bmp filetype, and must be in the same directory as the executable. If the file cannot be found, no splash is displayed and no error is indicated.

This means that you no longer need to rebuild the runtime executable file to customize the splash screen. Instead, you just put bitmap file and an executable file with the same name in the same directory.

For specific details on providing a custom splash screen, refer to the chapter on customizing the splash screen in the *Smalltalk User Guide*.

Migrating control files

Previous versions of VisualAge Smalltalk used control files (.ctl) to identify the configuration maps that make up a loadable feature. Beginning in VisualAge Smalltalk V6.0, configuration maps became the recommended method of loading features. Configuration maps are easier to maintain and update than control files. Configuration maps and their relationships are also controlled permanently and centrally within the library manager. For information on using configuration maps for loading features, see the *Visual Programming User Guide*.

Additionally, VisualAge V6.0 supports pre-importing feature code libraries at install time. This helps reduce problems when loading features and eliminates the need to install feature *.dat files separately. VisualAge Smalltalk V6.0 provides a packaged command-line utility to import feature code. This utility is included with every client or manager install. See Importing feature code.

There are still some situations, however, where control files work better than configuration maps. VisualAge Smalltalk V6.0 and VA Smalltalk will continue to support control files, in a modified way.

The following table shows the different parts of a control file and what is supported in VisualAge Version 6.0.

Control file information	Support in Version 6.0 and VA Smalltalk
header information	Supported with changes
include	Supported
includeFeatureMap	New in Version 5.5
import	Not supported
mapname	Supported with changes

header

The header is used to specify the feature name and an expression that is evaluated to determine if the feature is valid for the specific image.

V6.0 and later syntax: `<feature name><commentQuotedBooleanExpression>`, where `<feature name>` is the string that the user sees in the Load Features dialog and `<CommentQuotedBooleanExpression>` is a Smalltalk expression enclosed in the Smalltalk comment quotes, `""`.

V6.0 and later syntax: `<visibility><kind><FeatureName><commentQuotedBooleanExpression>`, where `<visibility>` is either `z.` to indicate the feature is visible in the Load Features dialog, or `zz.` to indicate that the feature is not visible, `<kind>` indicates whether the feature is a VA Smalltalk feature or a Smalltalk Base feature. For VA Smalltalk features, use `VA:`; for Smalltalk Base features, use `ST:`.

The `<commentQuotedBooleanExpression>` works as it did in V5.0. If the expression evaluates to true, the control file is considered valid for the current image. If the expression evaluates to false, the feature definition will be ignored for this image. The content of the expression has changed; however. In V6.0 and later you provide a new expression: `"AbtCommonProductInstallerApp validPlatformIds: 'woahs'"`. The parameter is a string of characters to identify the operating systems where the feature is supported. The valid values for this method are w for windows, a for AIX, l for Linux and s for Solaris, in any order.

For example, the header for a visible feature that is valid on Windows and AIX would look similar to the following:

```
z.ST: My Smalltalk Feature "AbtCommonProductInstallerApp validPlatformIds:
'aw'"
```

include

You can include other control files that are prerequisites for this feature.

Syntax: `include=abtebd50.ctl`

includeFeatureMap

You can include the name of feature configuration maps in the control file. This allows you specify another feature as a prerequisite.

Syntax: `includeFeatureMap=zz.ST: My Smalltalk Feature`

import

You no longer need to import feature code during the feature load. Feature code is pre-imported using the Library Importer Tool.

mapname

The mapname is used to specify the name and version of a configuration map to be loaded.

Syntax: `<mapname><versionSpec>`, where `mapname` is the name of the configuration map to be loaded and `<versionSpec>` is either an exact timestamp (specification of the form ts=1234556789, where 1234556789 is the integer number-of-seconds in the map's timestamp), or a version name pattern. The version name pattern can include wildcards. If a version name pattern is used, the chosen edition will be the newest edition whose version name matches the pattern.

The import library name and comment used in Version 5.0 are no longer used.

Examples:

```
'mymap' 'ts=3131946603'
'my other map' 'v7.8.6.3'
'yet another map' 'v7.8.*'
```

Examples:

Here are two examples of Version 6.0 control files:

```
z.ST:Sample Parts "xxxCommonProductInstallerApp validPlatformIds:'woa'"
include=xxxeq55.ctl

'My Sample Parts' 'ts-3033647458'
```

This example is for a visible Smalltalk feature. The control file information also indicates that the valid platforms for this feature are Windows (w), OS/2 (o), and AIX (a). The feature includes the configuration maps from another feature, specified in the `xxxeq55.ctl` file. There is one configuration map associated with this feature, specified by the map name `'My Sample Parts'` and timestamp.

```
z.VA:Sample Parts "true"
includeFeatureMap=zz.ST: My Smalltalk Feature
'My Sample Parts' 'Sample Parts 1.0'
```

This example shows that this control file is for a visible VisualAge feature. The control file is always valid, since the boolean is true. This control file includes a configuration map from another feature, specified in the map `zz.ST: My Smalltalk Feature`. The `zz` at the beginning of the map name indicates

that this feature is not shown in the feature load/unload list. This control file also has a configuration map associated with it, specified with the map name and version.

Importing feature code

All feature code must be imported into the manager at install time. This is true whether a feature is loaded via the new feature-loading maps or via the older control files. VA Smalltalk installs a Library Importer Tool in the <vast>\importer directory of machines with the client or manager product installed.

Windows: Library Importer Tool is called `importer.exe`.

Linux: Library Importer Tool is called `importer`.

The Library Importer Tool accepts the following parameters:

-z.target=
accepts the location of the target library. This is the customer's current code library. The library can be specified in one of two ways:

libraryPath

This will use fileIO to access the library.

server::libraryPath

This will use EMSRV to access the library.

-z.source=
accepts the location of the source import library. The format is the same as for the target.

-z.sourcedir=
accepts a directory path. Code will be copied from of each library (i.e. each file with a DAT extnsion) in the directory.

Note:

Either `-z.source` or `-z.sourcedir` must be specified, but never both.

-z.silent
Runs with no user interface (no longer used as of V8.6.3; it will be ignored)

-z.allVersions
Accepts ‘true’ to import all versions of configuration maps in the source library or ‘false’ to import only the newest version of configuration maps in the source library (default is ‘true’).

After the tool runs, check for non-zero exit codes. The Library Importer Tool will end with a 0 (zero) exit code if the import was successful. If the import was not successful, the tool will end with a non-zero exit code.

The following are examples of how to run the Library Importer Tool :

- `importer.exe -z.target=zot::d:\vast1201\manager\mgr1201.dat -z.source=e:\tmp\myfeat.dat`
- `importer.exe -z.target=zot::d:\Program_Files\12.0\manager\mgr1201.dat -z.source=e:\tmp\myfeat.dat`

- `importer.exe -z.target=zot::d:\vast1201\manager\mgr1201.dat -z.sourcedir=e:\tmp\featureDir`

Note:

With File I/O only one person may be using the code library at any one time or you will get the following error message:

```
Error 65: Open Failed.
```

Migrating method mappings to JDK 1.2

In order for Smalltalk to use RMI with JDK 1.2, Smalltalk must know the method hash information. If you are using JDK 1.1, the method hash value is ignored.

The mapping migration utility searches the image for existing JDK 1.1 mappings and gives the user an option to update these to new JDK 1.2 mapping. Run the migration utility by loading the SstRmiMigration application and executing the following code:

```
SstRmiMigration runMigration
```

For more information regarding RMI and JDK, please see the *VA Smalltalk Server Smalltalk Guide*

Changes in Server Workbench

Smalltalk Server, the runtime environment for deploying server applications, is included with VisualAge Smalltalk Server Workbench in VisualAge Smalltalk V6.0 and VA Smalltalk. This means that all of the code required to deploy server applications is included as part of Server Workbench and it is no longer necessary to install this feature separately.

The deployment environment for Server Workbench is not backward compatible with Server v5.0 or earlier. This means that applications packaged on Server Workbench V5.0 or earlier will not run on the redistributable files for Server Workbench V6.0 or VA Smalltalk. To enable your V5.0 or earlier application to run on a V6.0 or VA Smalltalk system, repackage your application using the current version of Server Workbench.

If your systems need to run applications packaged from both releases, keep Server Runtime V5.0 installed in parallel with the current version of Server Runtime on those systems.

Current code page support is inconsistent on supported platforms

Beginning with VisualAge Smalltalk V6.0, the *currentCodePage* method answers a String on Unix and MVS platforms and an Integer on Windows platforms. Thus, the current code page conversion

support is inconsistent on supported platforms. Several of the code page conversion routines use the *currentCodePage* as an argument.

This *currentCodePage* inconsistency causes problems for applications that must execute on multiple platforms. For example, a user who passes *String* code pages to Windows conversion routines will encounter a walkback because the Windows conversion API expects Integers. UNIX accepts either Integers or Strings, but the 'iconv' API must be passed a String code page name in order to work properly.

AbtConnectionSpec >>#targetCodePage: expects a String argument

Beginning with VisualAge Smalltalk V5.5, the argument passed to the *AbtConnectionSpec >>#targetCodePage:* method is expected to be a String. The argument was previously expected to be a numeric value. This change was implemented as part of the XML Support introduced with V5.5.

Migrating from Version 5.5

This part addresses concerns for all existing users of VisualAge Smalltalk V5.5 who are migrating to the current release of VA Smalltalk current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Elimination of required map links in configuration maps
- Low-level configuration map reorganization
- Time class shape changed
- Default XML serialization methods added for Date and Time
- Changed behavior of millisecondClockValue on OS/390
- Behavior change in handling negative system time
- Behavior change in creating FileStreams
- Packager 'do not reduce' rule behavior changed
- EMSRV startup parameter changes
- SST HTTP Servlet API
- POSIX Migration on OS/390
- Behavior changes for Application Builder parts

Elimination of required map links in configuration maps

Most of the configuration maps shipped in VisualAge Smalltalk V6.0 and later (with the exception of those maps identifying loadable features) have been changed to eliminate their required map links. If you load an image from maps, you will need to either select and load the supplied maps in the correct order or load using the feature definition maps (for example, `z.ST: FeatureName`).

Low-level configuration map reorganization

The low-level ENVY configuration maps were reorganized in VisualAge Smalltalk V6.0. If you have required map links from your maps to any of the obsoleted maps, you will need to change them since the obsoleted maps are not shipped with VA Smalltalk.

VisualAge Smalltalk V5.5.2 and earlier:

- ENVY/Image (now obsolete)
- ENVY/Image Batch Runtime (changed)
- ENVY/Image Batch Runtime Extensions (now obsolete)

- ENVY/Image Batch Runtime Platform Support (now obsolete)

VisualAge Smalltalk V6.0 and later:

- ENVY/Image Base
- ENVY/Image Batch Runtime
- ENVY/Image Compiler
- ENVY/Image Development
- ENVY/Image Graphical User Interface

The *ENVY/Image* map was refactored by moving all development time applications to the *ENVY/Image Development* map, the Smalltalk compiler applications to the *ENVY/Image Compiler* map, and the GUI applications to the *ENVY/Image Graphical User Interface* map.

The *ENVY/Image Batch Runtime* map was refactored to remove all applications that were included in other maps (specifically *ENVY/Image Base*).

The applications that were in the *ENVY/Image Batch Runtime Extensions* and *ENVY/Image Batch Platform Support* maps were duplicates of what is included in the *ENVY/Image Base* map.

Time class shape changed

In order to increase the precision of *Time*, an instance variable, *milliseconds*, has been added and the *secondsSinceMidnight* instance variable has been renamed to *millisecondsSinceMidnight* in VisualAge Smalltalk V6.0. This change should be transparent to existing applications since the API for *Time* was not changed, but only extended.

The object loader mutation method for *Time* has been updated to account for the difference in class shape. This will make loading instances of *Time* that were dumped by a previous version of VisualAge Smalltalk work correctly as long as you **did not** specify the optimization of not including variable names when you created the dumped file (the default behavior is for variable names to be included).

Default XML serialization methods added for Date and Time

Date and Time objects are constructed automatically for XML strings in the format specified by ISO 8601. This was done to enable automatic conversion for the 'date' and 'time' types of the XML schema. Users can create custom methods to convert to and from other formats if desired. The XML mapping specification must be updated to enable usage of these custom methods. Specify an 'AttributeClassCreationMethod' in the XML mapping specification to enable a custom method for converting an XML string to a Smalltalk object. Specify an 'ObjectToStringConversionMethod' to enable a custom method for converting a Smalltalk object into a suitable XML string.

Changed behavior of millisecondClockValue on OS/390

Beginning with VisualAge Smalltalk Server for OS/390 and z/OS V6.0, the *millisecondClockValue* method returns a value based on the STCK assembler instruction, which gives the current value of the time-of-day clock (TOD). For versions prior to VisualAge Smalltalk Server for OS/390 and z/OS V6.0, the *millisecondClockValue* method returned the application CPU usage time.

Behavior change in handling negative system time

Prior to VisualAge Smalltalk V6.0, the design assumed that if the system time goes backwards, then the system's tick counter must have rolled over to zero. However, there are at least 3 situations that can cause time to go backwards:

1. The tick counter rolls over to zero
2. Daylight Savings Time (or Summer Time) ends
3. A hardware or software clock synchronizer program makes a negative adjustment to the time

By treating all 3 of these conditions the same, errors were introduced in code that compares successive probes of the system's tick counter. For simplicity, situations 1 and 2 will continue to be treated alike. This introduces (or rather, doesn't eliminate) a once-a-year error.

Situation 3 will be treated as a no-op condition by introducing a time epsilon. Code taking successive probes of the system's tick counter using methods such as *millisecondClockValue* or *microsecondClockValue* should make use of this epsilon value (obtained from the *CldtConstants* pool dictionary entries *MillisecondClockValueEpsilon* or *MicrosecondClockValueEpsilon*) to effectively ignore small negative adjustments in the tick counter. For an example of this usage, see the *Time class>>#millisecondsToRun:* method.

The default values for negative time epsilon are 5 minutes.

Behavior change in creating FileStreams

In previous releases, *FileStream class>>#write:*, *FileStream class>>#write:mode:*, and *FileStream class>>#write:mode:check:type:* created instances which support read-write filestreams. However, according to the ANSI Smalltalk standard, they should create instances that support writeFileStream protocol only. Since these methods were introduced for ANSI compatibility, it is only reasonable that they should perform according to the standard. The effect of this change is that it is no longer valid to use any of the readFileStream protocol with a filestream object created using these methods.

Packager 'do not reduce' rule behavior changed

The "do not reduce" rule in packager did not include the methods from the specified class in the computing of the transitive closure. As such, it could produce a non-working image with messages being sent to methods which have been "accidentally" reduced out. This often, but not always, showed up as errors when running the **Examine & Fix Problems** step of the packager.

So, for example, if there is a class 'A' that defines these methods:

```
#main (class method)
^super new method1

#method1
| a |
a := Array new: 10.
^a

#method2
| a |
a := Bag new.
^a
```

When packaging a normal "runtime reduced image" with startup code 'A main', then these methods would be included:

```
A class>>#main
A>>#method1
```

Also, the Array class would be included in the packaged image.

If a packaging rule like the following were added:

```
packagingRulesFor: aPackagedImage
"Define rules for the given packaged image."

aPackagedImage doNotReduceClassNamed: self name
```

and the packager rerun, then these methods would be included:

```
A class>>#main
A>>#method1
A>>#method2
```

But, the class Bag would not be included (as the user might expect) even though it is referenced by #method2. The reason is that the reduction algorithm only considered "explicitly" included methods to be starting points for reduction. Implicitly included methods were effectively added after the runtime reduction - i.e. runtime reduction is done as always and then the remainder of any methods/classes/apps which have "do not reduce" rules are added.

The implementation of the 'do not reduce' rule has been changed such that all the methods in the unreduced components are used as seeds in the reduction algorithm.

EMSRV startup parameter changes

There have been several changes in the startup parameters for EMSRV. These are detailed in the EMSRV section of the *Installation Guide*.

SST HTTP Servlet API changed

SST support for the Sun Servlet web application model has been updated. SST now closely models the major features of Servlet API 2.2. As a result, there have been many SST API changes. Servlet protocol has changed, and a model of *HttpServletRequest* has been introduced. In general the result is a significantly extended API, but some legacy API has now been marked obsolete.

The classes *SstHttpCgiRequestHeader* and *SstHttpCgiServlet* are now obsolete. Existing references to *SstHttpCgiRequestHeader* should be changed to make use of *SstHttpRequestHeader* instead. Existing references to *SstHttpCgiServlet* should be changed to make use of *SstHttpServlet* instead. An implementation of these obsolete classes is provided, to facilitate migration only, in the application *SstObsoleteCgiCompatibility*.

The class *SstHttpServer* has a new role. It now represents "a server that understands the HTTP protocol"; it specifically knows nothing about servlets. The role of "servlet container" is now filled by *SstHttpServletEngine*.

Setting up a simple web application is quite different beginning with VisualAge Smalltalk V6.0, as it is based on the model of a web application provided by the Servlet 2.2 API. The primary conceptual difference is that one now must build a declarative web application specification object and deploy this in a web application server, whereas previously one specified the content of the web application by using API on the server directly.

The legacy example http server (*SstHttpServerExample class>>#runAt:in:*) should be reviewed as a guide for defining and deploying a web application. This example has been extended to demonstrate some specific features of the new API, such as cookies, sessions, and error signaling.

POSIX Migration on OS/390

Prior to VisualAge Smalltalk Server for OS/390 and z/OS V6.0, the Delay timer was not implemented in the Smalltalk VM running on OS/390 and z/OS. Therefore all Smalltalk processes were forced to run at priority 3 and all wait message sends to Delay converted to a *Processor>>#yield*. This behavior was the same whether the LE POSIX runtime option was set ON or OFF.

Beginning with VisualAge Smalltalk Server for OS/390 and z/OS V6.0, the Delay timer is implemented when running Smalltalk with the LE POSIX runtime option set to ON in the Native batch environment (the Delay timer is not implemented when running under IMS or CICS). With POSIX set ON the *ProcessorScheduler* runs as described in the *Programer's Reference*. The priority is not forced to a

priority of 3 and the wait message send is not converted to a *Procesor>>#yield*. With the LE POSIX runtime option set to OFF the behavior is the same as if the Delay timer is not implemented.

Prior to VisualAge Smalltalk Server for OS/390 and z/OS V6.0, all asynchronous calls were converted to synchronous blocking calls. Beginning with VisualAge Smalltalk Server for OS/390 and z/OS V6.0, in a Native Batch environment with the LE POSIX runtime option set ON and the *EsAsynchronousCallout* application included in the image, all asynchronous calls run on separate operating system threads. In the IMS and CICS environments the asynchronous calls will continue to be converted to synchronous blocking calls.

Behavior changes for Application Builder parts

In the course of adding feature enhancements in VisualAge Smalltalk V5.0, inadvertent changes were made to existing behavior. We have restored the original behavior as detailed below. For many customers these changes will not affect their current applications.

Container Details Column: For a container details column with the following properties set:

- converter = String
- emptyMakesDefault = true
- defaultObjectFromEmptyString= <a blank character>

the column will no longer force you to highlight and delete the blank character before entering a new value.

Container Details Column: For container details columns the zero suppressed setting did not correctly suppress the zero in all cases. To have the zero appear change zero suppressed from true to false.

TextConverter: For converters that have the property *emptyMakesDefault* set to *true* and the *defaultObjectFromEmptyString* set to *nil,* you will no longer get a change event signaled when the value goes from *nil* to *nil*.

Migrating from Version 6.0

This chapter addresses concerns for users of VisualAge Smalltalk V6.0 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- AbtPortableNotebookPageView inconsistent behavior
- ScaledDecimal>>#hash changed
- Subtle change to Compiler
- Using the 'Class Variable Initializer' packaging rules for IC packaging
- Removed unused method #abtXmlPrintOn:namespaces:
- Suppress serialization of nil attribute values when attribute mapping specifies Required=false

AbtPortableNotebookPageView inconsistent behavior

In VisualAge Smalltalk V6.0.3, code was changed in AbtPortableNotebookPageView so that invoking the *aboutToBeSwitchedTo* event was not deferred. This aligned its implementation with *aboutToBeSwitchedFrom* which also is not deferred. Before this change was made, if a user quickly clicked on more than one notebook tabs which had associated events, the events could be run out of order. For example, if Page2 has the *aboutToBeSwitchedTo* and *aboutToBeSwitchedFrom* events connected, and if a user is on Page1 and rapidly clicks first on the tab for Page2 and then on the tab for Page3, the events could be run in the order *switchFromPage2* followed by *switchToPage2*. This is obviously incorrect behavior.
Unfortunately, some user applications depended on the deferral of execution of the *aboutToBeSwitchedTo* event. For example, it could be used to set focus to a particular subpart of the notebook page when the tab for the page was selected. These application were broken by the change.
Since it is not possible to automatically satisfy the needs of both sets of applications -- those that depend on execution deferral and those that depend on no execution deferral -- another solution is needed.
One possible solution for these broken applications is to change their implementation so that the *aboutToBeSwitchedTo* handler does the deferral of execution itself if needed. So a method that needs this capability and that used to look like this:

```
aboutToBeSwitchedTo: aComp

 aComp setFocus
```

could be rewritten as:

```
aboutToBeSwitchedTo: aComp
```

```
[aComp setFocus] abtDefer
```

Another solution is to return the system's implementation to the old (deferral of execution) behavior. This can be done by changing the the value of following line in the .INI file from *false* (VA Smalltalk default behavior) to *true* (VisualAge Smalltalk V6.0.2 and previous behavior):

```
[Version Compatibility]
deferPortableNotebookPageSwitchToCallback=false
```

ScaledDecimal>>#hash changed

The algorithm used to hash a ScaledDecimal object changed in VisualAge Smalltalk V6.0.1. This change affects existing collections that rely on the hash value. The affected collections are, at least,

- `KeyedCollection` (and its subclasses) and
- `Set` (and its subclasses).

If your application uses `ScaledDecimal` objects as keys for any of these hashed collections, you **must** recompute the hash values after installing the current version of VA Smalltalk and before modifying the collection in any way. To recompute the hash values, send the `rehash` message to the collection for example,

```
myDictionaryKeyedWithScaledDecimals rehash.
```

Subtle change to Compiler

Due to a subtle change to the compiler (`CodeStream>>#literalIndexFor:`), some methods, when recompiled, may behave differently in VA Smalltalk than they did in earlier versions of VisualAge Smalltalk.

The original problem involved an error in the way the compiler determined whether 2 array literals were identical when the array literals contained one or more numeric values (see `badArrayIdentity` below as an example).

This problem was corrected in VisualAge Smalltalk V6.0.1, but some unforeseen side-effects were introduced as shown in the table below. These side-effects were eliminated in VisualAge Smalltalk V6.0.2.

While testing the changes for VisualAge Smalltalk V6.0.2, another (pre-existing) problem in this same code was discovered and fixed (see `poolDictionaryModification` below as an example).

```
badArrayIdentity
 | a |
 a := #(3.0).
 ^a == #(3) "Should answer false"

stringLiteralIdentity
 | a |
 a := 'hello'.
 ^a == 'hello' "Should answer true"
```

```
mixedArray
 |a|
 a := #('string' #symbol 3).
 ^a == #('string' #symbol 3) "Should answer true"

poolDictionaryModification
 "Upon method compilation, both constants = 'y'"
 "Upon entry to this method, TestPoolB::Value1 = 'y'"
 TestPoolA::Value1 := 'x'.
 ^TestPoolB::Value1 "Should answer 'y'"
```

	V6.0 and earlier	6.0.1	V6.0.2 and later
badArrayIdentity	true	false	false
stringLiteralIdentity	true	false	true
mixedArray	true	false	true
poolDictionaryModification	'x'	'x'	'y'

Using ‘Class Variable Initializer’ packaging rules for IC packaging

Previous to VisualAge Smalltalk V6.0.1, class variable values were not saved when packaging ICs. The `#initializeClassVariable:to:inObjectNamed:` and `#initializeClassVariable:toObject:inClassNamed:` packaging rules were being ignored. In VA Smalltalk, these packaging rules have their desired effect of initializing the identified class variable(s) in the packaged runtime IC.

You should examine the senders of these messages in your code to ensure that they specified the initialization values that you want for the class variables at runtime.

Removed unused method #abtXmlPrintOn:namespaces:

In VisualAge Smalltalk V6.0.2, the following public methods are removed:

- AbtXmlSchema>>#abtXmlPrintOn:namespaces:
- AbtXmlSchemaComplexType>>#abtXmlPrintOn:namespaces:
- AbtXmlSchemaDeclaration>>#abtXmlPrintOn:namespaces:

These methods were not used by supported versions of the VAST XML support and should not affect user applications.

Suppress serialization of nil attribute values when attribute mapping specifies Required=”false”

This change does NOT affect objects that utilize an XML schema for serialization. Nil attribute values are now suppressed during serialization of attributes with attribute mappings that specify 'Required="false"'. For example:

```
<ClassElementMapping ElementTagName="customer"
 ClassName="AbtXmlSampleCustomer" >
<AttributeMapping ClassAttribute="lastName" Required=”false”>
   <Attribute>lastName</Attribute>
</AttributeMapping>
 </ClassElementMapping>
```

If the *lastName* attribute of a serialization target object is nil, the serializer will not render the *lastName* attribute.

Migrating from Version 7.0

This chapter addresses concerns for users of VA Smalltalk V7.0 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Windows theme support interferes with some testing tools
- SST HTTP Server access log

Windows theme support interferes with some testing tools

In previous versions of VisualAge Smalltalk and VA Smalltalk, the CommonWidgets framework instantiated several custom window classes to provide additional functionality. Some of these custom window classes interfered with Windows theme support.

Change

In VA Smalltalk V7.5, the custom windows classes were removed and only standard windows classes are now used. The removed custom window classes are OSBUTTON (now BUTTON) and TEXTEDIT (now EDIT).

Action Required

This change may cause problems with automated GUI testing tools (such as Mercury WinRunner, HP Quick Test, HP Unified Functional Test) that depend on knowing the window class of widgets. If you experience failures of automated tests due to this change, you will need to modify your tests to reference the standard window classes in place of the custom classes.

SST HTTP Server access log

Previous versions of VisualAge Smalltalk and VA Smalltalk automatically wrote an access log showing all incoming requests. This was not always a desirable action, especially at runtime.

To obtain the access log in the current version of VA Smalltalk, you need to start your development or run time image using the -sstaccesslog command line switch.

Migrating from Version 7.5

This chapter addresses concerns for users of VA Smalltalk V7.5 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Modifiable splash screen on Windows
- Consolidate EtToolsVendorExtensionsApp andVisualAgeVendorExtensionsApp
- Server Smalltalk (SST) feature definition changes
- Merge ENVY/Image Controls apps into apps
- Changes to AbtTimestamp class
- New conversion and testing API methods
- Manifest file changes some GUI appearance on Windows
- Merge AbtCwControlsEdit and AbtCwControlsRun apps into other maps
- Change to SstHttpServletRequest>>#getContentType

Modifiable splash screen on Windows

Beginning with VA Smalltalk V7.5.2, the product splash bitmap is built into the Windows executable. If you install over a previous release, you need to delete the `<varoot>\bin\abt.bmp` file so that the built-in splash will be used. Please refer to ‘Customizing the icon and splash screen’ in the *Smalltalk User Guide* for further details of this new support.

Consolidate EtToolsVendorExtensionsApp and VisualAgeVendorExtensionsApp

The vendor extension applications were provided to allow add-on tool providers to extend the Smalltalk browser menus. Historically, since they were not provided with the Smalltalk product itself, the vendor extension applications were included with each add-on tool. This caused a problem since there were multiple incompatible versions of the vendor extension applications available and included. Beginning with VA Smalltalk V7.5.1, the newest version of the *EtToolsVendorExtensionsApp* has been included in the *ENVY/Image Development* map and the newest version of the *VisualAgeVendorExtensionsApp* has been included in the *VisualAge Development Environment* map. Add-on tool providers should remove these applications from their distribution packages to ensure the vendor extensions are not back-leveled.

Server Smalltalk (SST) feature definition changes

In VisualAge Smalltalk and VA Smalltalk V7.5.2 and earlier, there were 2 features defined for Server Smalltalk (SST):

- ST: Client, SST
- ST: Server, SST

Generally, the idea was that the *ST: Client, SST* feature included those maps needed to build a Server Smalltalk Client application and the *ST: Server, SST* feature included those maps needed to build a Server Smalltalk server application using the Cross Development Environment (XD). Unfortunately, this idea did not match the actual content of the features – both features were incorrect in their content.

For the current version of VA Smalltalk, the *ST: Client, SST* feature has been replaced with a suite of features and the *ST: Server, SST* feature has had its content reduce to only what is needed to support loading Server Smalltalk into the XD environment and subsequent packaging from that environment.

The new features replacing the *ST: Client, SST* feature are:

- ST: Server Smalltalk (SST) – Base
- ST: Server Smalltalk (SST) – CORBA IIOP
- ST: Server Smalltalk (SST) – Distributed Objects
- ST: Server Smalltalk (SST) – HTTP
- ST: Server Smalltalk (SST) – Java RMI
- ST: Server Smalltalk (SST) – Mail (IMAP/SMTP)
- ST: Server Smalltalk (SST) – Message Queuing (MQ)
- ST: Server Smalltalk (SST) – Seaside
- ST: Server Smalltalk (SST) – Seaside Testing
- ST: Server Smalltalk (SST) – Web Services

To support this reorganization of the Server Smalltalk features, changes were needed to the underlying configuration maps. This is a purely organizational change, no applications were added or deleted. The changes are:

- Server Smalltalk - Basic Tools (added)
- Server Smalltalk - Distributed Examples (added)
- Server Smalltalk - Distributed Tools (added)
- Server Smalltalk - HTTP Examples (added)
- Server Smalltalk - IIOP Examples (added)
- Server Smalltalk - Java RMI Examples (added)
- Server Smalltalk - Java RMI Tools (added)
- Server Smalltalk - Mail Examples (added)
- Server Smalltalk - Packaging Examples (added)
- Server Smalltalk - Tools (deleted)
- Server Smalltalk - Wizard Tools (deleted)

Merge ENVY/Image Controls apps into other maps

ENVY/Image Controls was a configuration map shipped in VisualAge Smalltalk and in VA Smalltalk V7.5.2 and earlier. It contained applications associated with Windows native controls (such as *TabStrip*, *ToolBar*, etc.):

- *CwControls*

- *CwControlExamples*
- *CwControlUseCases*

Change

In VA Smalltalk V8.0 and later, these apps have been made subapplications and therefore are included in other maps:

- *CwControls* became *CwWindowsControls* subapplication of *CommonWidgets*
- *CwControlExamples* became *CwWindowsControlsExamples* subapp of *CwExamples*
- *CwControlUseCases* became *CwWindowsControlsUseCases* subapp of *CwUseCases*

Action required

The effect of this change is twofold:

- if any of your maps have *ENVY/Image Controls* as a required map link, the link should be changed to *ENVY/Image Graphical User Interface* or *ENVY/Image Examples* instead
- if any of your apps have *CwControls, CwControlExamples*, or *CwControlUseCases* as prerequisites, the prerequisite should be changed to *CommonWidgets*, *CwExamples*, or *CwUseCases* instead (CwUseCases is in the *ENVY/Image Interactive Test Suites* map).

Changes to the AbtTimestamp class

Reason for change

AbtTimestamp, our perennial problem-child, has undergone yet another class shape change to alleviate a serious performance problem.

Change

The 2 instance methods *timestampValue* and *timestampValue:*, which were categorized as AbtBase-Internal, were none the less used in some customer code. These methods have now been categorized as AbtBase-Deprecated. *timestampValue* is replaced by *asMicroseconds*; *timestampValue:* is replaced by the class method *fromMicroseconds:*. These 2 new methods are categorized as AbtBase-API and so should remain stable in the future.

The instance method *milliseconds:* is also now categorized as AbtBase-Deprecated without a direct replacement.

The behavior of the class method *date:time:microseconds:* has been changed to better align with the internal representation of *AbtTimestamp*. Previously the value of the *microseconds:* argument could range from 0 – 999999 and overrode the milliseconds portion of the *time:* argument. Now the valid range is 0 – 999 and it does not affect the milliseconds portion of the *time:* argument.

Action required

If you want to set the milliseconds portion of a timestamp, use code similar to the following:

```
anAbtTimestamp time milliseconds: anInteger
```

Since the methods categorized as AbtBase-Deprecated may be removed from the product in some future release, you should consider changing your code so that these methods are not used.

New testing and conversion CLDT API methods

Several new methods categorized as API have been added to the VA Smalltalk *Kernel* application. Some of the methods are convenience methods designed to reduce the need for using the *isKindOf:* method (which is expensive).

It is possible that one or more of these methods may conflict with a similar (or identical) method provided by your code. In this case, you need to remove the method from your code.

- *asString* (moved from *WbKernel* application)
- *isBlock*
- *isCollection*

It is very likely that additional *isKindOf:* replacement methods will be added in the future.

Manifest file changes some GUI appearance on Windows

The inclusion of a manifest file (such as `abt.exe.manifest`) defers decisions about some GUI appearance elements to Windows. Specifically, shortcut keys are underlined without a manifest, but with a manifest, the underlining is controlled in the operating system, and the default is not to underline shortcut keys.

To change the default underlining of shortcut keys on Windows XP:P

- open the control panel;
- choose “Display”
- select the Appearance tab;
- click the Effects button;
- the resulting dialog has a check box for "hide underlined letters for keyboard navigation until I press the Alt key"; it should not be checked;
- dismiss the dialogs with OK.

To change the default underlining of shortcut keys on Windows Vista or Windows 7:

- open the control panel;
- choose "Ease of Access Center";
- choose "Make the keyboard easier to use";
- scroll down to "Underline keyboard shortcuts and access keys" and check it;
- click Save.

Merge AbtCwControlsEdit and AbtCwControlsRun apps into other maps

AbtCwControlsEdit and *AbtCwControlsRun* were configuration maps shipped in VisualAge Smalltalk and in VA Smalltalk V7.5.2 and earlier. They contained development time and runtime applications associated with Windows native controls (such as *TabStrip*, *ToolBar*, etc.):

- *AbtEditWinCwControlsApp*
- *AbtRunWinCwControlsApp*

Change

In VA Smalltalk V8.0 and later, these apps have been made subapplications and therefore are included in other maps:

- *AbtEditWinCwControlsApp* was merged into *AbtWinEditViewsSubapp* subapplication of *AbtEditViews*
- *AbtRunWinCwControlsApp* was merged into *AbtWinRunViewsSubapp* subapplication of *AbtRunViews*

Action required

The effect of this change is twofold:

- If any of your maps have *AbtCwControlsEdit* as a required map link, the link should be changed to *VisualAge Development Environment* instead; if any of your maps have *AbtCwControlsRun* as a required map link, the link should be changed to *VisualAge for Smalltalk – Run UI* instead.
- If any of your apps have *AbtEditWinCwControlsApp* as a prerequisite, the prerequisite should be changed to *AbtEditViews* instead; if any of your apps have *AbtRunWinCwControlsApp* as a prerequisite, the prerequisite should be changed to *AbtRunViews* instead.

Change to SstHttpServletRequest>>#getContentType

Reason for change

SstHttpServletRequest>>#getContentType did not conform to the definition of this interface in *Java Servlet Specification 2.4, SRV.14.2.16.1, getContentType()* which states that the method should return “a String containing the name of the MIME type of the request, or null if the type is not known.”

Prior to V8.0, *SstHttpServletRequest>>#*getContentType returned the entire contents of the Content-Type field of the header (for example `text/html; charset=ISO-8859-4`) rather than just the MIME type (for example `text/html`).

Change

SstHttpServletRequest>>#getContentType now returns just the MIME type (for example `text/html`).

Action required

If your application expects to receive the MIME type and parameters (such as `charset=ISO-8859-4`) from *SstHttpServletRequest>>#getContentType*, you will need to modify your application to account for this change.

Migrating from Version 8.0

This chapter addresses concerns for users of VA Smalltalk V8.0 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Change to the UNIXProcess class
- Change to default RMI stubProtocolVersion
- Change to exception handling
- Change to Dictionary>>#copy behavior
- AbtWaitApp moved from IBM Smalltalk, ALL - UI Run to IBM Smalltalk, ALL - Headless configuration map
- New testing CLDT API methods
- Object>>#initialize method added

Changes to the UNIXProcess class

Reason for change

In VisualAge Smalltalk and VA Smalltalk 8.0 and earlier, *UNIXProcess* always used the `csh` shell executable. This shell executable is not the normal one on Linux, and is not even available on some Linux standard installs.

Change

UNIXProcess now allows the user to set a class variable, *CurrentShell,* to specify which shell executable to use. The shell executable should reside in (or be linked into) the `/bin` directory. The default is either the shell specified by the environment variable `SHELL`, or `bash` if this variable is not set.

Action required

If you depended on the default being `csh`, you will need to explicitly set the environment variable `SHELL` or the *UNIXProcess* class variable *CurrentShell*.

Change to default RMI stubProtocolVersion

Reason for the change

SstRmiMarshalingConfiguration>>#stubProtocolVersion has defaulted to *SstRmiStubVersion1* since the RMI support was first introduced into VisualAge Smalltalk. In JDK 1.2.2, a new version of RMI stubs was introduced. The SST RMI support was updated to handle this new version, but the default was never changed; rather, the documentation was updated to explain that the user should reset the RMI stub protocol version if running on JDK 1.2.2 (or later).

Change

The default *SstRmiMarshalingConfiguration>>#stubProtocolVersion* has been changed to *SstRmiStubVersion2*.

Action required

If you need to use Version 1 stubs, you should refer to the *Server Smalltalk Guide* for information on how to set the stub version.

Change to exception handling

Reason for the change

In VisualAge Smalltalk and VA Smalltalk 8.0 and earlier, if an exception occurred while execution an *Exception*'s *defaultAction* or an *ExceptionalEvent*'s *defaultAction*, there were no exception handlers available to handle the exception since the chain of exception handlers had been followed to the end before executing the *defaultAction*.

Additionally, if another exception occurred in a protected block after an exception handler had been executed, that exception might not have been handled properly since some or all of the chain of exception handlers had been consumed.

Change

The exception handling code has been updated to restore the chain of exception handlers before executing the *defaultAction* for an exception and before resuming or exiting from an exception handler.

Action required

Since the exception handling behavior has changed in situations where multiple exceptions occur in a protected block (or nested protected blocks), you should evaluation your exception handling strategy to ensure this change does not adversely affect your application.

Change to Dictionary>>#copy behavior

Reason for the change

The definition of the *copy* operation is: “Return a new object that must be as similar as possible to the receiver in its initial state and behavior. Any operation that changes the state of the new object should not,

as a side-effect, change the state or behavior of the receiver. Similarly, any change to the receiver should not, as a side-effect, change the new object.”

The implementation of *Dictionary>>#copy* did not conform to this definition.

Change

The implementation of *Dictionary>>#copy* has been changed to conform to the definition.

Action required

You should examine the usage of *Dictionary>>#copy* in your applications to ensure that you have no dependency on the copy of a *Dictionary* being updated when the original *Dictionary* is updated. If you do, you will need to update your applications to use *#shallowCopy* in place of *#copy* in these cases.

AbtWaitApp moved from IBM Smalltalk, ALL - UI Run to IBM Smalltalk, ALL - Headless configuration map

Reason for the change

Although there is no code in the *AbtWaitApp* that is dependent on a GUI, its subapplication configuration expressions referred to a subsystemType of ‘CW’. This prevented it from being loaded into a Cross Development (XD) image since the ‘CW’ subsystemType is not set in XD images.

Change

The *AbtWaitApp* subapplication configuration expressions have been changed to refer to a subsystemType of ‘OS’. Since it has no GUI dependency, the *AbtWaitApp* application was moved from the IBM Smalltalk, ALL – UI Run configuration map to the IBM Smalltalk, ALL – Headless configuration map.

Action required

No action is required. When creating a Cross Development (XD) image, the *AbtWaitApp* application will be loaded automatically into that image.

New testing CLDT API methods

Several new methods categorized as API have been added to the VA Smalltalk *Kernel* application. These methods are convenience methods designed to reduce the need for using the *isKindOf:* method (which is expensive).

It is possible that one or more of these methods may conflict with a similar (or identical) method provided by your code. In this case, you need to remove the method from your code.

- *isArray*
- *isAssociation*
- *isBehavior*
- *isDictionary*
- *isFraction*
- *isNumber*
- *isPoint*

It is likely that additional *isKindOf:* replacement methods will be added in the future.

Object>>#initialize method added

Reason for change

Some ported open source applications contain subclasses of *Object* that send *super initialize* in their own initialization code.

Change

Add an empty *Object>>#initialize* method.

Action required

This change does not directly affect the function of any code.

If any of your applications supply an empty *#initialize* method in subclasses of *Object*, you can (but don't have to) remove the method.

What this change does affect is the results of static code analysis tools such as ENVY/QA Code Critic or Smalllint. They may indicate a possible bug that was not previously indicated. Here is the description from Code Critic for this possible bug:

```
Should call superclass
 Warns if the checked method specializes selectors
 but does not call their superclass implementation. For
 example, when implementing #initialize it is common
 to call super #initialize.
```

Adding *super initialize* to *#initialize* methods which are flagged by these tools is an appropriate change.

Migrating from Version 8.0.1

This chapter addresses concerns for users of VA Smalltalk V8.0.1 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Grease Core and Grease Core Tests maps are obsolete
- EsString>>#subStrings: changed to be ANSI-compliant and portable
- Random class and methods referring to it have been removed
- SequenceableCollection>>#beginsWith: and #endsWith: have been replaced
- Duration>>#milliseconds changed to be portable
- IdentitySet class added as replacement for EsIdentitySet
- Color class promoted from WbKernel to Kernel
- SortedCollection private methods #sorted and #sorted: renamed

Grease Core and Grease Core Tests maps are obsolete

Reason for change

The *Grease Core* and *Grease Core Tests* maps were replaced with the more descriptive *Platform Portability* and *Platform Portability Tests* maps, but the old maps were not removed from the shipped manager.

Change

Grease Core and *Grease Core Tests* have been removed from the shipped manager.

Action required

If you have required map links to *Grease Core* or *Grease Core Tests*, you need to change them to refer to *Platform Portability* or *Platform Portability Tests*. You should purge the obsolete maps from your manager so there is no confusion in the future regarding their usage.

EsString>>#subStrings: changed to be ANSI-compliant and portable

Reason for the change

The implementation of *EsString>>#subStrings:* did not conform to the definition of this method in the *ANSI Smalltalk Standard* (which states that the method should take a collection of separator characters as its argument).

In addition, its behavior in some situations (leading, trailing, and multiple adjacent separator characters) did not match the behavior of *EsString>>#subStrings* nor did it match the behavior of the *#subStrings:* method on other Smalltalk platforms (which makes porting code from other platforms to VA Smalltalk difficult).

EsString>>#subStrings: parsed strings as follows:

```
'@b' subStrings: $@   answered #('' 'b')
'@' subStrings: $@    answered #('' '')
'abc' subStrings: $@  answered #('abc')
'' subStrings: $@     answered #('')
```

Change

Three changes have been made:

EsString>>#subStrings: will now accept either a single character separator or a collection of character separators as its argument. It will parse strings as follows:

```
'abc@def:123' subStrings: $@    answers #('abc' 'def:123')
'abc@def:123' subStrings: '@'   answers #('abc' 'def:123')
'abc@def:123' subStrings: '@:'  answers #('abc' 'def' '123')
```

EsString>>#subStrings: will now ignore leading separators, trailing separators, and separators which are immediately adjacent to other separators. It will parse strings as follows:

```
'@b' subStrings: $@   answers #('b')
'@' subStrings: $@    answers #()
'abc' subStrings: $@  answers #('abc')
'' subStrings: $@     answers #()
```

EsString>>#allSubStrings: has been added. It behaves the same way that *EsString>>#subStrings:* did in previous releases.

Action required

The change to *EsString>>#subStrings:* which makes it ignore leading separators, trailing separators, and separators which are immediately adjacent to other separators may require changes to user code (depending on the exact expectations of senders of the message).

Here are several examples of code along with descriptions of how they should be changed:

1. Splitting a directory path into its component parts:

```
('/home/user/bin/command' subStrings: $/) reject: [:string | string = ''].
```

2. While it is no longer necessary to scan the results of applying *#subStrings:* to the path in order to remove empty strings from the result, it causes no harm and does not need to be modified.Splitting a directory into its component parts and referencing a component part by its index in the results:

   ```
   ('/home/user/bin/command' subStrings: $/) at: 2.
   ```

   Since *#subStrings:* no longer answers an empty string for a leading separator character, this code must be modified. It can be changed in at least 2 different ways to achieve the desired result:

   ```
   ('/home/user/bin/command' subStrings: $/) at: 1.

   ('/home/user/bin/command' allSubStrings: $/) at: 2.
   ```

3. Splitting an XLFD font name into its component parts:

   ```
   ('-microsoft-system-bold-r-normal--16-100-120-120-p-90-iso8859-1'
      subStrings: $-) size = 15.
   ```

   Notice that an XLFD font name has a fixed number of component parts (15) and can have null elements (look just after `normal`), so it is not sufficient simply to change the code to test for a size of 14. Rather, the code must be changed to use *#allSubStrings:* so that 15 component parts are still created.

This change may also affect third-party tools, such as HP WinRunner (an automated functional GUI testing tool), also. If you use such a tool, you should contact the tool vendor to determine whether the tool needs to be updated.

Random class and methods referring to it have been removed

Reason for change

The *Random* class and the methods referring to it (*#atRandom* and *#atRamdom:*) in *GreaseVASTCoreApp* are obsolete

Change

GreaseCoreApp now provides a platform-independent random number interface, so the *Random* class and the methods referring to it (*#atRandom* and *#atRamdom:*) have been removed.

Action required

Change:

```
aCollection atRandom
```

to:

```
GRPlatform current newRandom randomFrom: aCollection
```

Change:

```
anInteger atRandom
```

to:

```
GRPlatform current newRandom nextInt: anInteger
```

SequenceableCollection>>#beginsWith: and #endsWith: have been replaced

Reason for change

The *#beginsWith:* and *#endsWith:* methods were provided as part of the Seaside portability code. It was then discovered that these methods are not cross-platform portable. They do not give the same result for an empty argument (for example, *''* or *#()*) on all platforms. Rather than attempt to get some platforms to change their implementation of these methods, it was decided to provide new methods that have the desired behavior on all platforms.

Change

SequenceableCollection now provides platform-independent methods *#beginsWithSubCollection:* and *#endsWithSubCollection:* which should be used in place of *#beginsWith: and #endsWith:* (which have been removed).

Action required

Change:

```
aCollection beginsWith: aSubCollection
```

to:

```
aCollection beginsWithSubCollection: aSubCollection
```

Change:

```
aCollection endsWith: aSubCollection
```

to:

```
aCollection endsWithSubCollection: aSubCollection
```

Duration>>#milliseconds changed to be portable

Reason for change

Other Smalltalk platforms answer just the millisecond portion of a *Duration* for the *#milliseconds* message.

Change

The implementation of Duration is changed to enhance cross-platform portability:

- the *#milliseconds* method is changed to answer just the millisecond portion of a *Duration*.
- the *#asMilliseconds* message is added to answer the value of a *Duration* expressed in milliseconds (the old behavior of *#milliseconds*).

Action required

Change:

```
aDuration milliseconds
```

to:

```
aDuration asMilliseconds
```

IdentitySet class added as replacement for EsIdentitySet

Reason for change

Other Smalltalk platforms provide an *IdentitySet* class with behavior identical to VA Smalltalk's *EsIdentitySet*. *EsIdentitySet* had no API methods, so was not guaranteed usable by customer applications. VA Smalltalk has provided *IdentitySet* as a public subclass of *EsIdentitySet* when the *Platform Portability* map is loaded.

Change

- The *IdentitySet* class is moved to the base with all the behavior of *EsIdentitySet*.
- All extensions to *EsIdentitySet* provided by VA Smalltalk are moved to *IdentitySet*.
- *EsIdentitySet* is made a subclass of *IdentitySet*. It has no behavior of its own and is considered a deprecated class.

Action required

Change any *EsIdentitySet* references in your applications to *IdentitySet*.

Color class promoted from WbKernel to Kernel

Reason for change

Seaside needs to be able to access the *Color* class to provide HTML color tags, possibly in a headless image. This class was only available when the *WbKernel* application (part of WindowBuilder Pro – Runtime) was loaded and when the *CommonGraphics* GUI Framework was available.

Change

- The *Color* class is moved to the base with all its original behavior than is not dependent on a GUI framework.
- The *CgRGBColor* class is made a subclass of *Color* to provide behavior based on having a GUI framework available.
- All previous *WbKernel* extensions to *Color* that depend on having a GUI framework are moved to *CgRGBColor*.
- Instances of *Color* support RGB color intensities of 0 – 255.

Action required

Change any *Color* references in your applications that provide values to a GUI framework to *CgRGBColor*.

SortedCollection private methods #sorted and #sorted: renamed

Reason for change

Seaside uses the *#sorted* and *#sorted:* methods to obtain sorted copies of collections. VA Smalltalk had private implementations of these methods on *SortedCollection* to facilitate lazy sorting.

Change

- *SortedCollection* is updated with new private methods *#isSorted* and *#isSorted:* implementing the previous behavior of *#sorted and #sorted:* (accessors for the *sorted* instance variable).
- *SortedCollection*'s implementation of *#sorted* and *#sorted:* is changed to match the expected cross-platform behavior (answers a sorted copy of the receiver).

Action required

Since these were private methods, this change should not affect any users. However, if your application subclasses *SortedCollection*, or sends *#sorted* or *#sorted:*, you may need to update your application.

Migrating from Version 8.0.2

This chapter addresses concerns for users of VA Smalltalk V8.0.2 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Object>>#value added to CLDT
- Icon management classes have been reworked
- Mastering ENVY/Developer Configuration Expression Prompter configuration map removed
- Changes to ScaledDecimal>>#printOn: and #printString
- Changes to TrailBlazer Browsers

Object>>#value added to CLDT

Reason for change

The primary reason for the change was the need of an implementation of Object>>#value by certain Seaside add-ons. However, in researching this need, it was also discovered that the *GF/ST Graph Layout* map already contained an *Object>>#value* extension, resulting in inconsistent behavior for Seaside add-ons, as well as for user applications, depending on whether or not the *GF/ST Graph Layout* map was loaded.

Change

Object>>#value has been added to CLDT returning *self*. Product implementations of *#value* on subclasses of *Object* which answered *self* have been removed.

Action required

If you have added *Object>>#value* as an extension in your application, you need to take one of the following actions:

- If your extension method simply answers *self*, you can remove the extension.
- If your extension method does anything but simply answer *self*, you will need to either move the method to a subclass of *Object*, or recode your application so that it does not rely on *Object>>#value* doing anything but answering *self*.

Icon management classes have been reworked

Reason for change

There has been a problem with leaking GDI resources on Windows due to *CgIcon* instances not being freed properly. The underlying problem was “white box” reuse of a bad implementation of an icon manager.

Change

A new hierarchy of icon managers has been provided based on the old implementation of *EwxIconManager* (which was the only correct implementation):

```
EwIconManager

EwButtonIconManager

WkIconManager

EwTriangleIconManager

WbIconManager

EwxIconManager
```

EwIconManager provides common behavior for its subclasses as well as the capability to manage individual icons.

EwButtonIconManager and *EwTriangleIconManager* provide specific support for tree hierarchy indicators.

WkIconManager (from *WidgetKit/Controls - Runtime*) and *WbIconManager* (from *WindowBuilderPro – Runtime*) have been deprecated and are empty classes kept only to satisfy any references to the classes in customer code.

EwxIconManager provides icon management specific to the *EwExamples* application.

EpIconManager (from *ENVY/Packager*) and *JLOIconManager* (from *SUnit Browsers*) have been removed since they exhibited the problem and provided no additional function over that provided with the new icon manager classes.

EwIconPolicy has 2 new convenience class methods available to create standard icon policies -- *#defaultTrianglePolicy* (for “twisty” hierarchical indicators) and *#defaultButtonPolicy* (for +/- button hierarchical indicators). This should reduce the need for explicit references to the *…IconManager* classes.

Action required

If you use, extend, or subclass any of the *…IconManager* classes, you may need to make changes to your application:

- If you have subclasses of any class named *…IconManager*, you should reevaluate the need for your subclass. If you believe your subclass is still needed, you should ensure that you are subclassing the correct class and that your changed or added function still works properly.
- If you have extended any class named *…IconManager*, you should reevaluate the need for your extension. If you believe your extension is still needed, you should ensure that you are extending the correct class and that your changed or added function still works properly.
- If your application makes direct reference to any class named *…IconManager*, you should reevaluate the need for the reference. If you believe a reference is still needed, you should ensure that the reference is made to the correct new class.

Mastering ENVY/Developer Configuration Expression Prompter configuration map removed

Reason for change

Users have requested that the browser extension provided by the *Mastering ENVY/Developer Configuration Expression Prompter* be included in the base product.

Change

The Configuration Maps Browser and the Application Editions Browser are updated by adding *Edit…* to the Config. Expression menu. In addition, the prompter used for *Add…*, *Edit…*, and *Copy…* has a multi-line entry field to better display complex configuration expressions. The VA Assist version of these browsers has similar updates.

Action required

If you have development maps which depend on the *Mastering ENVY/Developer Configuration Expression Prompter* map, change the dependency to *EtTools*.

Changes to ScaledDecimal>>#printOn: and #printString

Reason for change

`ScaledDecimal>>#printOn:` produced output that included the scale value (i.e., `23.1s3 printOn: aStream` produced ’23.1s3’). Since the #`printString` method uses `#printOn:` to provide its output, `23.1s3 printString` also produced ’23.1s3’. This made `#printString` unusable for producing customer readable values.

Change

`ScaledDecimal>>#printOn:` was changed so that it no longer produces the scale value in its output (i.e., `23.1s3 printOn: aStream` and `23.1s3 printString` now both produce ’23.1’.

Action required

If you have a dependency on the previous output format for `ScaledDecimal>>#printOn:` (or `#printString`), you will need to change your code to use `#printOn: <aStream> showDigits: nil pad: false` instead.

Changes to TrailBlazer Browsers

Reason for change

When the *TrailBlazer* feature was loaded, it changed the current browser set to point to itself. In addition, the Tools menu items used to enable and disable the TrailBlazer browsers referred to the TrailBlazer browsers as *enhanced browsers*, a somewhat confusing reference given the capabilities of the VA Assist browsers.

Change

When the *TrailBlazer* feature is loaded, it does not change the current browser set.

The non-standard Tools menu items **Use Enhanced Browsers** and **Use Standard Browsers** menu toggles have been consolidated into a single **Use TrailBlazer Browsers** menu toggle.

Because of these changes, the message written to the Transcript when the *TrailBlazer* feature is loaded is changed from:

```
The TrailBlazer browser is now the default browser.
To make the standard browsers the default, use
the ''Use Standard Browsers'' option from the
Smalltalk tools pulldown on this window.
```

to:

```
The TrailBlazer browsers have been loaded but are
not the default browsers.  To make them the default
browsers, select the ''Use TrailBlazer Browsers'' option
from the Tools pulldown on this window.
```

As part of this update, the content of the *TrailBlazerNlsSupportApp* application has been merged into the *TrailBlazer* application.

Action required

If you have a previous version of the *TrailBlazer* application loaded into your image, you will need to unload it and then the *TrailBlazerNlsSupportApp* applications before loading this version of the *TrailBlazer* application.

If you want to use the TrailBlazer browsers, select **Use TrailBlazer Browsers** from the Transcript's Tools pulldown menu after loading the *TrailBlazer* feature.

Migrating from Version 8.0.3

This chapter addresses concerns for users of VA Smalltalk V8.0.3 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Consolidation of EsbBenchmarksES subapplication
- Change Float and ScaledDecimal 32-bit hash for better distribution
- INI file changes introduced by Preference Settings Framework
- Change to CgScreen class>>bitmapPath and Locale class>>nlsPath

Consolidation of EsbBenchmarksES subapplication

Reason for change

This subapplication is left over from the time when ENVY/Stats supported VisualWorks.

Change

All code from *EsbBenchmarksES* has been merged into *EsbBenchmarks*.

Action required

- If you package the benchmarking code into a runtime application, the *#actualIncludedSubapplications* method of the packaging instructions will change the next time you run the Packager.

Change Float and Scaled Decimal 32-bit hash for better distribution

Reason for change

The current result of sending *abtHash32* to a *Float* or a *ScaledDecimal* has a very bad distribution:

- 2 *Float* or *Scaled Decimal* objects whose only difference is to the right of the decimal point hash to the same value
- Large *Float* or *ScaledDecimal* objects all hash to the same value
- The signs of *Float* or *Scaled Decimal* objects are ignored when calculating their hash value

Change

Float>>abtHash32: and *ScaledDecimal>>abtHash32:* have been implemented. These methods treat the *Float* or *ScaledDecimal* like a *ByteArray* (possible because their internal representation is a *ByteArray*), thus giving a more uniform distribution of hash values.

Action required

If your application uses *Float* or *ScaledDecimal* objects as keys for any keyed collection (such as *AbtHighCapacityDictionary*) that uses *abtHash32* to calculate the hash values of its keys, you **must** recompute the hash values for that collection after installing the current version of VA Smalltalk and before modifying the collection in any way. To recompute the hash values, send the *rehash* message to the collection. For example:

```
myDictionaryKeyedWithScaledDecimals rehash.
```

INI file changes introduced by Preference Settings Framework

Reason for change

A new Preference Settings Framework is available in this release. This framework is optimized to process settings from the `INI` file contained in stanzas with names matching the names of the Applications that the settings apply to. This means that some reorganization of existing entries in the `INI` file is needed.

Additionally, the .INI file syntax uses the (; - semicolon) character both for the comment delimiter and for the array of values separator (for example, in the bitmapPath= setting). This makes it impossible to tell the difference between an array of values and a single value followed by a comment.

Change

The following table shows the reorganization changes made in the `INI` file.

Old Stanza	Old Keyword	New Stanza	New Keyword
Kernel	fplevel	AbtCommonProductInstallerApp	*same*
	EvaluationLogName	EtBaseTools	*same*
	AllowOutOfScopeReferences	*deleted – unused*	
NLS Config	ignoreCatalogAliasing	AbtNlsKernelApp	*same*
VAST 5.0 Compatibility	installationPath	AbtCommonProductInstallerApp	*same*

There are likely to be more changes in the future as additional setting entries in the `INI` file are moved from custom settings handler processing to Preference Settings Framework handler processing.

The separator between elements in an array of values is changed to be the (, - comma) character.

The following applications have been removed since they provided the (now-unused) preference setting support used before introduction of the new Preference Settings Framework:

- *AbtImageConfigurationApp*
- *AbtEditImageConfigurationApp*

Action required

If you maintain your own customized `INI` files, they should be updated as indicated above.

If you have applications that rely on the previous (undocumented) preference setting support in the 2 removed applications, you should convert your code to use the new Preference Settings Framework as described in the *Smalltalk User Guide*.

Change to CgScreen class>>bitmapPath and Locale class>>nlsPath

Reason for change

Previously there was no standardization for the representation of directory path strings that were retrieved from the INI file. Sometimes these strings were stripped of their trailing path separator when saved, sometimes a path separator was added when they were stored, and sometimes they were stored as entered (for example, from the `Common Graphics bitmapPath=` setting in the INI file).

Change

The Preference Settings Framework introduced in this release standardizes the representation of directory path names to include the trailing directory separator character. This affects all directory path preferences read from the INI file using the new Preference Settings Framework:

- [CommonGraphics] bitmapPath=
- [CommonPrinting] printPath=
- [NLS Config] nlsPath=
- [VAST 5.0 Compatibility] installPath=

Most of these INI file settings are normally used only by internal VA Smalltalk routines. However, user applications may have references to these setting values:

- `CgScreen class>>bitmapPath` will now always answer an array of directory names with a trailing directory separator. There will be no empty strings (‘’) in the array.
- `Locale class>>nlsPath` will now always answer an array of directory names with a trailing directory separator. There will be no empty strings (‘’) in the array.

Action required

You should examine your application code for senders of the methods mentioned above. Because of the inconsistency in the way directory names were previously handled, such references will often include code to add a missing directory separator before appending a filename to the directory name. Therefore, code that looks like this:

```
Locale nlsPath do: [:aPath | | path |
  path := (aPath notEmpty and: [ aPath last = CfsDirectoryDescriptor
 pathSeparator ])
```

```
      ifTrue: [ aPath, aFilename ]
      ifFalse: [ aPath, CfsFileDescriptor pathSeparatorString, aFilename ].
    "Code that does something with aPath" ]
```

Can be changed to this:

```
Locale nlsPath do: [:aPath | | path |
  path := aPath, aFilename.
  "Code that does something with aPath" ]
```

You should also example your application code for any code that sets either of these values (senders of `Locale class>>nlsPath:` or `CgScreen class>>bitmapPath:`). Ensure that the arguments to these setters are correctly constructed (Array of directory names with trailing directory separator, no empty entries).

Migrating from Version 8.5

This chapter addresses concerns for users of VA Smalltalk V8.5 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Changed specification of ODBC dll or shared object in .INI file
- Renamed PlatformConstants::Error to PlatformConstants::Errorregion for Windows
- Added 2 Parameters to the socketAppender Setting in the ini File
- Changed default settings for VA Assist Pro
- AbtHover Tooltip text now left-aligned by default
- Class definition template changed in standard browsers

Changed specification of ODBC dll or shared object in .INI file

Reason for change

Mapping of logical names to physical names for dlls and shared objects is being consolidated into the [PlatformLibrary Name Mappings] stanza of the .INI file.

Change

The *ODBCLibraryName* keyword is moved from the [Database] stanza to the [PlatformLibrary Name Mappings] stanza of the .INI file.

Action required

If you have specified the name of an ODBC library dll (Windows) or shared object (Unix) by including the [Database] stanza with the *ODBCLibraryName* keyword in your .INI file in order to access a custom ODBC library, you should move line containing the *ODBCLibraryName* keyword to the [PlatformLibrary Name Mappings] stanza (replacing the existing *ODBCLibraryName* line that is already there). If the [Database] stanza is empty after making this change, you can remove it.

Changed PlatformConstants::Error to PlatformConstants::Errorregion for Windows

Reason for change

PlatformConstants::Error conflicts with the ANSI Exceptions *Error* class. This made it impossible to refer to the *Error* exception class in methods of any class specifying the *PlatformConstants* pool dictionary on Windows without resorting to the convoluted `System image globalNamespace classAt: #Error` idiom.

Change

PlatformConstants::Error was renamed to *PlatformConstants::Errorregion* on Windows. This aligns it with the 3 other region return codes (*Nullregion*, *Simpleregion* and *Complexregion*). All code shipped with VA Smalltalk that referred to *PlatformConstants::Error* was updated.

Action required

- *Error* was (and *Errorregion* now is) a return code from a small number of calls into Windows from methods in the *CgRegion* class. If you have code that referenced the *Error* entry in *PlatformConstants*, you need to change the reference to be *Errorregion*.
- If you have classes that both include the *PlatformConstants* pool and contain methods that refer to the *Error* class, you will need to touch those methods and resave them so they bind *Error* to the class rather than the *PlatformConstants* entry.

Added 2 Parameters to the socketAppender Setting in the ini File

Reason for change

The log4s socket appender will now attempt to retry the connection if the connection cannot be made when the image starts.

Change

- The socketAppender line in the ini file has two new parameters:
- retryCount - the number of times the connection will be retried. Zero means never retry.
- retrySeconds - the numbers of seconds to wait between retry attempts.
- The new syntax is socketAppender=(someCaseSensitiveAppenderName, someCaseSensitiveLoggerName , levelName, patternLayout, patternString, ipAddress, hostName , port, retryCount, retrySeconds)

Action required

Add two non-negative integer values as the last two parameters for socketAppender.

Example: socketAppender=(TestSocketAppender, root, All, EsPatternLayout, '%m %c', '127.0.0.1' , localHost, 4433, 99, 30)

Changed default settings for VA Assist Pro

Reason for change

The out of the box behavior for our IDE may be different from what you are used to. The default settings for VA Assist Pro were changed to be compatible with the new Code Assist feature and to remove dependency on default screen color depth for Unix platforms

Change

The new default setting for Unix is

- UseColorSyntax = true

The setting changed to support Code Assist.

The new default settings for both Windows and Unix platforms are:

- CheckSpelling = true.
- DragDrop = false
- UseEditBars = true
- UseEnhancedListWidgets = true
- UseToolBars = true

Action required

No action is required if you are used to working in the VA Smalltalk IDE with the above settings.

If these settings are irritating, Chapter 16 "Tools Menu Enhancements" in the *VA Assist Pro User Guide* tells how to change these settings.

Warning:
If you turn spell checking off, code assist will produce a walkback when requesting method completions.

Unix:
if you set UseEditBars to false the IDE will not work. You will get walkbacks in some situations, for example, when you ask for implementers or senders of a method from the transcript.

Abt Hover Tooltip text now left-aligned by default

Reason for change

In most implementations, the multi-line text within hover tooltips is left-aligned to improve readability. No setting was previously specified in VA, and so prior to 8.5.1 the tooltip would default to using center-aligned text.

Change

AbtHoverHelpManager will provide the new left-alignment specification when creating the hover tooltip label.

Action required

If you are using multi-line tooltips and would prefer center-aligned text, you can update the alignment for the tagLabelWidget, which is accessible through the AbtHoverHelpManager instance that is managing your tooltip enabled widgets.

Example:

```
| hoverHelp hoverHelpAdapter hoverHelpManager |

hoverHelpAdapter := hoverHelp hoverHelpAdapterForWidget: self widget.

hoverHelpManager := hoverHelpAdapter hoverHelpManager.

hoverHelpManager tagLabelWidget alignment: XmALIGNMENTCENTER
```

Class definition template changed in standard browsers

Reason for change

The original implementation of VisualAge Smalltalk did not support class instance variables. When this capability was added (many years ago), the class definition template used by the standard browsers was not updated. In an image *without* VA Assist loaded, the class definition template was:

```
NameOfSuperclass subclass: #NameOfClass

instanceVariableNames: 'instVarName1 instVarName2'

classVariableNames: 'ClassVarName1 ClassVarName2'

poolDictionaries: ''
```

but in an image *with* VA Assist loaded it was:

```
NameOfSuperclass subclass: #NameOfClass

classInstanceVariableNames: 'classInstVarName1 classInstVarName2'

instanceVariableNames: 'instVarName1 instVarName2'

classVariableNames: 'ClassVarName1 ClassVarName2'

poolDictionaries: ''
```

Change

To reduce confusion, and to enable identifying class instance variables without editing the template, the class definition template used in standard browsers is changed to match the one with VA Assist loaded.

Action required

If any of your application code is sensitive to the content of the class definition template, you may need to update this code to account for the new class definition template.

Migrating from Version 8.5.1

This chapter addresses concerns for users of VA Smalltalk V8.5.1 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Parameters in the [log4s] stanza of the .INI file should not be enclosed in parentheses
- Time Zone support introduced
- Prerequisite ICs changed in many IC packaging instructions
- The String>>asInteger method has been removed from EsLoggingFrameworkApp

Parameters in the [log4s] stanza of the .INI file should not be enclosed in parentheses

Reason for change

The initial release of log4s did not use the Preference Settings Framework, and used a non-standard way of supporting multiple parameters by enclosing them in parentheses.

Change

When using the Preference Setting Framework, parentheses are used to indicate array values, not multiple simple values.

Action required

Examine the entries in your [log4s] stanza. Remove the parentheses surrounding the values.

Example:

```
socketAppender=(TestSocketAppender, root, All, EsPatternLayout, '%m %c',
'127.0.0.1', localHost, 4433, 99, 30)
```

should be

```
socketAppender=TestSocketAppender, root, All, EsPatternLayout, '%m %c',
'127.0.0.1', localHost, 4433, 99, 30
```

Time Zone Support Introduced

Reason for change

VA Smalltalk did not correctly handle calculations between two *DateAndTime* objects when they span a Daylight Savings Time transition.

Change

The *DateAndTime* object was changed to use time zone data.

Action required

When repackaging existing VAST Reduced RunTime Images, you must manually include the application EsTimeZoneApp.

Changes in the DateAndTime class have been implemented such that code written before the implementation of time zones will continue to execute properly without changes.

Likewise, DateAndTime objects stored before the time zone capability was implemented are seamlessly mutated when they are read into a Smalltalk image which supports time zones.

Use of deprecated APIs to create DateAndTime precludes calculations using Daylight Savings Time transition.

- DateAndTime class>> year: year day: dayOfYear hour: hour minute: minute second: second millisecond: millisecond offset: aDuration
- DateAndTime class>> year: year month: month day: dayOfYear hour: hour minute: minute second: second millisecond: millisecond offset: aDuration

See “Time zones” in the *Smalltalk User Guide* for more information.

Prerequisite ICs changed in many IC packaging instructions

Reason for change

An application included in the *EpCfsNlsInstructions* IC packaging instructions has new application prerequisites which affect the prerequisite ICs for cfsnls.ic.

Change

EpCfsNlsInstructions is updated to include *EpPlatformSupportInstructions* in its list of prerequisite ICs (class method *#prerequisiteICs*). All IC packaging instructions that include *EpCfsNlsInstructions* are updated to include *EpPlatformSupportInstructions* immediately before *EpCfsNlsInstructions* in their list of prerequisite ICs (class method *#prerequisiteICs*).

Action required

If you package your ICs using the **Packager Control Panel**, you should repackage and save your packaging instructions (if they changed).

If you package programmatically, or by using a tool other than the **Packager Control Panel**, you should examine and correct (if needed) the *#prerequisiteICs* class method in your packaging instructions classes as described above.

String>>asInteger has been removed from EsLoggingFrameWorkApp

Reason for change

This method was introduced in 8.5 in the EsLoggingFrameworkApp application for internal use only. It can cause conflicts with user code.

Change

This method has been removed from EsLoggingFrameWorkApp.

Action required

If you have changed your code to rely on this new method, you should remove all references to it. You can find all references to it by browsing all implementors of asInteger.

Migrating from Version 8.5.2

This chapter addresses concerns for users of VA Smalltalk V8.5.2 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- EsLogManager class>>error: and EsLogManager class>>info: have been removed from EsLoggingFrameWorkApp
- Signal>>#description changed for compatibility with class-based Exceptions
- EsTimeZone class has changed

EsLogManager class>>error: and EsLogManager class>>info: have been removed from EsLoggingFrameWorkApp

Reason for change

These two methods overrode methods in a super class. *EsLogManager class>>error:* prevented subclasses from signaling an error. *EsLogManager class>>info:* prevented *EsLogManager* from being subclassed.

Change

These two methods have been removed from *EsLoggingFrameworkApp* in favor of *logError:* and *logInfo:*. Other similar class side methods in *EsLogManager* have been deprecated (see Deprecation).

Action required

If your logging code uses *EsLogManager class>>info:* or *EsLogManager class>>error:*, you must change the code to *EsLogManager class>>logInfo:* and *EsLogManager class>>logError:* respectively. If your logging code uses any of the other deprecated methods, you should, but do not have to, change them as indicated in Deprecation.

Signal>>#description changed for compatibility with class-based Exceptions

Reason for change

In VA Smalltalk, class- and instance-based exceptions should be able to be used interchangeably. This implies that the semantics of methods defined separately for class- and instance-based exceptions should be the same. *Signal>>#description* and *Exception>>#description* did not conform to this rule.

For instance-based exceptions, *Signal>>#description* answered the constant description string of the *ExceptionalEvent* that was set when the exception was created; for class-based exceptions, *Exception>>#description* answered either <className> or <className>': '<messageText> depending on whether or not *#messageText* answered *nil*.

Change

Both class- and instance-based exceptions now answer a consistent value for *#description* -- <ee-description> or <ee-description>': '<messageText> depending on whether or not *#messageText* answers *nil*.

<ee-description> is the string answered when sending *exception description* to an instance of *Signal* or any of its subclasses, and is the same string that was previously answered by *Signal>>#description*.

Notice that the string answered by *#description* differs both from the previous class- and instance-based exception implementations.

Examples

The following examples show the message string produced by the *#description* method before and after this change.

```
[ (Array with: 1) at: 2 ]
  when: ExError do: [ :ex |
      Transcript cr; show: ex description.
      ex exitWith: nil ]
```

Before: (ExError) An error has occurred.
After: (ExError) An error has occurred.: Primitive failed in: Object>>#at: due to Index out of range in argument 1

```
[ self error: 'Could not do that because 15 screws are missing in the left
 wing' ]
  on: ExError do: [ :ex |
    Transcript cr; show: ex description.
    ex exitWith: nil ]
```

Before: (ExError) An error has occurred.
After: (ExError) An error has occurred.: Could not do that because 15 screws are missing in the left wing

```
[ Object  foo ]
  on: ExError do: [ :ex |
```

```
Transcript cr; show: ex description.
ex exitWith: nil ]
```

Before: MessageNotUnderstood: Object class>>foo
After: (ExMessageNotUnderstood) An exception has occurred: Object class does not understand foo

Action required

It is unlikely that any action is required since the change is only to the exact format and content of the string answered by *#description*. However, some customers have implemented complex logic to handle the different values answered by *#description* and/or *#messageText* and this logic can now be either removed or significantly simplified.

- If your application is dependent on the exact format or content of the string that *Signal>>#description* previously returned, then you should update your application to send *exception description* rather than *description* to an instance of *Signal* or alternatively, adapt your application to the new string format and content.
- If your application is dependent on the exact format or content of the string that *Exception>>#description* previously returned, then you should adapt your application to the new string format and content.

EsTimeZone class has changed

Reason for change

The DateAndTime object keeps track of its time zone by holding a timeZone instance variable, which is an EsTimeZone object. EsTimeZone objects -- and therefore DateAndTime objects -- can be fairly memory intensive because they contain not only a timeZoneRule instance variable indicating the current time zone rule, but also a timeZoneRuleSet instance variable providing a complete set of all the time zone rule for the time zone. For some time zones, the number of rules could run into the hundreds, making them larger than absolutely necessary and slowing database transactions.

Change

The timeZoneRuleSet instance variable in EsTimeZone has been replaced by a timeZoneName instance variable, which holds the name of the time zone, e.g., America/New_York, for the DateAndTime object. The entire time zone rule set can be retrieved if need be by using the time zone name. The timeZoneRule instance variable is unchanged.

Action required

None. DateAndTime objects that were created prior to 8.6 are mutated automatically to the new EsTimeZone shape and all the DateAndTime and EsTimeZone APIs remain the same.

Migrating from Version 8.6

This chapter addresses concerns for users of VA Smalltalk V8.6 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Deprecate and replace classes AbtBase64Coder and SstBase64Coder
- Deprecate and replace incorrect Windows code page conversion API wrapper methods
- Object>>#triggerEvent: added to AbtCLDTAdditions for portability
- Print methods added to Integer and Float in CLDT for portability

Deprecate and replace classes AbtBase64Coder and SstBase64Coder

Reason for change

There are many different Base64 encoding formats defined (see Base64 on Wikipedia, RFC 4648: The Base16, Base32, and Base64 Data Encodings and RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part 1: Format of Internet Message Bodies). The format supported by both `AbtBase64Coder` and `SstBase64Coder` classes most closely resembles MIME Base64 Content-Transfer-Encoding, but the encoding is not exactly correct. The 2 classes provide the same function – neither provides any unique function.

In addition, other encoding formats are needed by VA Smalltalk applications.

Change

A new application – `EsBase64CoderApp` – has been created to contain the classes supporting the different Base64 encoding formats. This application contains the following classes:

```
Base64Coder

   Base64CoderMime

   Base64CoderUrl

   Base64CoderXmlNmToken

      Base64CoderXmlName
```

This new application is included in the `ENVY/Image Base` configuration map and in the `kernel.ic` Image Component (IC).

The `AbtBase64Coder` and `SstBase64Coder` classes are deprecated. Their constructor methods (`AbtBase64Coder class>>#current` and `SstBase64Coder class>>#new`) will now answer an instance of `Base64CoderMime` since it provides the function closest to what these 2 classes provided. All other methods in these 2 classes have been removed.

The Base64 Encoding standards describe the returned value from a decoding operation as a string while the current `AbtBase64Coder` and `SstBase64Coder` classes answer a `ByteArray`. The new Base64 classes will answer a `String` as the result of a decoding operation in accordance with the standards.

Action required

Encoding

If you have specific references to `AbtBase64Coder` and/or `SstBase64Coder` in your applications, you should consider the environment where you are using the Base64 encoding:

- `Base64CoderMime` should be used for encoding elements of email messages
- `Base64Coder` should be used in almost all other instances
- `Base64CoderUrl`, `Base64CoderXmlNmToken`, and `Base64CoderXmlName` should be used in special situations where the normal set of encoding characters is inappropriate

Once you have chosen the class appropriate to your environment, replace `AbtBase64Coder current` or `SstBase64Coder new` with `Base64Coder<type> current`.

Decoding

Instances of `Base64Coder` and `Base64CoderMime` can be used interchangeably to decode standard-, new Mime-, and old (deprecated) Mime-encoded data. However, data encoded using `Base64CoderUrl`, `Base64CoderXmlNmToken`, and `Base64CoderXmlName` must be decoded using the same class that was used for its encoding since the encoding character set is different for each of them (and also different from the encoding character set of `Base64Coder` and `Base64CoderMime`).

If the senders of `#decode:` in your application expect (and need) a `ByteArray`, you should send `#asByteArray` to the result of the `#decode:` operation.

Other

You should add `EsBase64CoderApp` as an explicit prerequisite to your applications that use the new Base64 coder classes.

Deprecate and replace incorrect Windows code page conversion API wrapper methods

Reason for change

The following methods wrapping Windows code page conversion API calls use incorrect signatures for the Windows API calls:

- `multiByteToWideChar:`
  `dwFlags:`

  `lpMultiByteStr:`

  `cchMultiByte:`

  `lpWideCharStr:`

  `cchWideChar:`
- `wideCharToMultiByte:`
  `dwFlags:`

  `lpWideCharStr:`

  `cchWideChar:`

  `lpMultiByteStr:`

  `cchMultiByte:`

  `lpDefaultChar:`

  `lpUsedDefaultChar:`

The error is that `cchMultiByte:` should be `cbMultiByte:` to indicate that the argument is a count of the number of bytes in the multibyte buffer, not a count of characters.

Change

New corrected methods wrapping the Windows API calls are provided:

- `multiByteToWideChar:`
  `dwFlags:`

  `lpMultiByteStr:`

  `cbMultiByte:`

  `lpWideCharStr:`

  `cchWideChar:`
- `wideCharToMultiByte:`
  `dwFlags:lpWideCharStr:`

  `cchWideChar:`

  `lpMultiByteStr:`

  `cbMultiByte:`

  `lpDefaultChar:`

  `lpUsedDefaultChar:`

The original methods are deprecated and categorized as PI-Obsolete.

Action required

If your application invokes either of the deprecated methods, you should change to use the replacement methods. You should also verify that you are supplying a count of bytes, not characters, for the `cbMultiByte:` argument to the methods.

Object>>#triggerEvent: added to AbtCLDTAdditions for portability

Reason for change

Both `GlorpVaPort` and `WbVsePortabilitySupport` applications supplied the `Object>>#triggerEvent:` extension method for portability of Pharo and VSE code. This caused a loading error when both applications are loaded.

Change

The `Object>>#triggerEvent:` method is moved to a common prerequisite application (`AbtCLDTAdditions`). The method is categorized as `AbtRun-Portability`.

Action required

If any user applications provide the mentioned extension method, evaluate them to see if it should be removed (implementation is the same) or renamed (implementation is different).

Print methods added to Integer and Float in CLDT for portability

Reason for change

Both `GreaseVASTCoreApp` and `WbVsePortabilitySupport` applications supplied the `Integer>>#printPaddedWith:to:base:` extension method for portability of Pharo and VSE code. This caused a loading error when both applications are loaded.

Change

The `Integer>>#printPaddedWith:to:base:` method is moved to `CLDT`. In addition, the closely related `Float>>#printPaddedWith:to:` and `Integer>>#printPaddedWith:to:` methods are moved from `GreaseVASTCoreApp` to `CLDT`. The methods are categorized as `ES-Portability`.

Action required

If any user applications provide the mentioned extension methods, evaluate them to see if they should be removed (implementation is the same) or renamed (implementation is different).

Migrating from Version 8.6.1

This chapter addresses concerns for users of VA Smalltalk V8.6.1 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Deprecate classes OSHostent, OSProtoent, and OSServent
- EmCompressor compression/decompression algorithm changed
- Move SequenceableCollection>>#swap:with: to CLDT for shared use by RBBrowserUIApp and SeasideCoreApp
- Dictionary now implements #= and #hash
- Multipart Forms Processing Now Removes Only the Trailing Cr/Lf Leaving Whitespace Intact At the End of the Content
- SSL Version 2 & 3 Protocols Have Been Deprecated
- OpenSSL Libraries are no longer distributed with VA Smalltalk

Deprecate classes OSHostent, OSProtoent, and OSServent

Reason for change

There are 2 versions of these classes on Windows – *OSHostent*, *OSProtoent*, and *OSServent* in the *WindowsPlatformFunctions* subapplication and *OSHostEnt*, *OSProtoEnt*, and *OSServEnt* in the *SocketCommunicationsInterface* application. There are no references to the *OSHostent*, *OSProtoent*, or *OSServent* classes.

Change

The *OSHostent*, *OSProtoent*, and *OSServent* classes are moved from the *WindowsPlatformFunctions* subapplication to the *WindowsObsoletePlatformInterface* subapplication. Extensions to these classes are moved from the *WindowsPlatformAccessors* subapplication to the *WindowsObsoletePlatformInterface* application also.

The *OSHostEnt*, *OSProtoEnt*, and *OSServEnt* classes are moved from the *SocketCommunicationsInterface* application to both the *SciComWIN* and *SciCommUNIX* subapplications since there are minor differences in the layout of the structures represented by these classes on Windows and UNIX.

The *SciHostEnt*, *SciProtoEnt*, and *SciServEnt* classes are moved from the *SciComWIN* subapplication to the *SocketCommunicationsInterface* applications since they are the same on both Windows and UNIX. These classes are removed from the *SciComUNIX* subapplication.

Action required

If your applications have any references to the *OSHostent*, *OSProtoent*, or *OSServent* classes, you should add the *SocketCommunicationsInterface* application to the prerequisites of your application; change the class references to *OSHostEnt*, *OSProtoEnt*, or *OSServEnt*; and adapt your application to use the accessor methods of these classes.

EmCompressor compression/decompression algorithm changed

Reason for change

When storing source code in the library, the source code is compressed to save space. The compression algorithm (*V1*) used by *EmCompressor>>#compress* is sensitive to both the platform it is running on and to the current locale. This means that it may be impossible to decompress some source code if the platform or locale is changed after the source code is saved.

Change

A new compression algorithm (*V2*) is introduced that is not affected by the execution platform or the current locale, and a matching new decompression function is added to the native and server library access primitive modules:

- Compression is always done with the new *V2* compression algorithm
- Decompression supports both the old *V1* format and the new *V2* format

Action required

In short, any image connecting to a manager with >= 8.6.2 code in it will need to use the new library primitives distributed with 8.6.2 or beyond. Older library primitives cannot properly decompress source created by the new library primitives.

On Windows these library primitives are called emntv50.dll and emsrv50.dll. On Unix they are libemntv50.dll and libemsrv50.dll. Copy these from >= 8.6.2 installations and replace the files with the same name found in < 8.6.2 installations. All versions of VA Smalltalk installed on the system should now have updated library primitives.

Detailed Description

The migration from the old compression algorithm to the new algorithm is, by definition, imperfect since there is no way to change old images or old primitives. However, with a little care there should be no problem.

Note:

In the following table, old means from an 8.6.1 or earlier release while new means from an 8.6.2 or newer release. Also, library access primitives means emntv50.dll and emsrv50.dll (Windows) or libemntv50.so and libemsrv50.so (UNIX).

	Library contains only V1 compressed source	**Library contains both V1 and *V2* compressed source**
Old image; Old library access primitives	Source is compressed as V1. V1 compressed source is decompressed	Source is compressed as V1. V1 compressed source is decompressed; **V2 compressed source is either not decompressed, is corrupted during decompression, or causes an exception (depending on the content)**
Old image; New library access primitives	Source is compressed as V1. V1 compressed source is decompressed	Source is compressed as V1. V1 and V2 compressed source is decompressed
New image; New library access primitives	Source is compressed as V2. V1 and V2 compressed source is decompressed	Source is compressed as V2. . 1 and V2 compressed source is decompressed
New image; Old library access primitives	Source is compressed as V2. V1 compressed source is decompressed; **V2 compressed source is either not decompressed, is corrupted during decompression, or causes an exception (depending on the content)**	Source is compressed as V2. V1 compressed source is decompressed; **V2 compressed source is either not decompressed, is corrupted during decompression, or causes an exception (depending on the content)**

Looking at the table, you can discern that old library primitives should **never** be used when accessing a library containing *V2* compressed source code. However, since the issue only manifests itself during decompression, there is no damage to the content of the library if old library access primitives are used. To help remind you of this, new images will report a warning on the Transcript when they start if they are connected to the library using old library primitives.

Move SequenceableCollection>>#swap:with: to CLDT for shared use by RBBrowserUIApp and SeasideCoreApp

Reason for change

SequenceableCollection>>#swap:with: was provided by the *GreaseVASTCoreApp* application and is used by various Seaside applications. This method is also used in the RBBrowserUIApp application, but the Refactoring Browser does not supply an implementation.

Change

The implementation of *SequenceableCollection>>#swap:with:* was moved from the *GreaseVASTCoreApp* application to the *CLDT* subapplication and categorize as *ES-Portability*.

Action required

If your applications provide an implementation of *SequenceableCollection>>#swap:with:*, it should be removed after ensuring that it provided the same functionality as the new implementation..

Dictionary now implements #= and #hash

Reason for change

A product use case surfaced that requires the ability to compare 2 dictionaries for equivalence. However, *Dictionary* inherited its implementation of #= and *#hash* from *Object* and these methods were identical to #== and *#identityHash*, thus testing for the 2 dictionaries being identical, not equivalent.

Change

Dictionary>>#= and *Dictionary>>#hash* have been added in *CLDT*. This implementation is inherited by all subclasses of Dictionary.

Action required

If your applications provide an implementation of *Dictionary>>#=* and/or *Dictionary>>#hash*, remove them after ensuring that they provided the same functionality as the new implementation.

In general, this change should not affect your application. However, if you have used a dictionary as a key in another dictionary, you will need to change this usage. The degenerate case is using a dictionary as a key in one of its own entries, like this:

```
| dictionary association |

dictionary := Dictionary new.

association := dictionary -> 'bar'.

dictionary add: association.
```

Before this change you could retrieve the only entry in the dictionary using `dictionary at: dictionary` because the hash of a Dictionary did not change based on its content. After the change, `dictionary at: dictionary` will raise *ExCLDTKeyNotFound* because the hash of the key (dictionary) is based on the content of the dictionary, and therefore changed between when it was added to the dictionary and when it was retrieved. The workaround for this case is to use an *IdentityDictionary* rather than a *Dictionary*.

Multipart Forms Processing Now Removes Only the Trailing Cr/Lf Leaving Whitespace Intact At the End of the Content

Reason for change

The Multipart Forms specifications require trailing whitespace to be retained.

Change

HttpMultipartContentAssembler>>assemblePartsFrom: onto: was modified to retain trailing white space and remove only the trailing cr/lf, which indicates the end of the content.

Action required

If your application depended upon all trailing whitespace being removed along with the trailing cr/lf at the end of the content, you will need to take action to remove it after multipart processing is complete.

SSL Version 2 & 3 Protocols Have Been Deprecated

Reason for change

These protocols have known and high-profile security vulnerabilities.

Change

Definitions for SSLv2 and SSLv3 remain in the image, however usage of these when configuring an SSL connection will result in a runtime error.

Action required

Modify any references to the following pool variables (from SciSslConstants) to a more secure protocol reference such as SSLv23 or any of the TLS prefixed protocols found in SciSslConstants. Contrary to the naming convention, SSLv23 actually favors TLS based protocols with a possible fallback to SSLv3 and is suitable for use.

Deprecated Protocol References: **SSLv2, SSLv2_client, SSLv2_server, SSLv3, SSLv3_client, SSLv3_server**

OpenSSL Libraries are no longer distributed with VA Smalltalk

Reason for change

The OpenSSL release cycle includes quick turn-around patch releases based on discovered security vulnerabilities which demand immediate attention due to its ubiquitous use on the web. While previous versions of VA Smalltalk shipped with these libraries, it was noticed that they became dated (insecure) even within VA Smalltalk's release cycle. Many customers making use of SSL/TLS functionality were forced to upgrade these libraries anyway.

Change

Windows OpenSSL and supporting binary changes:

- **(removed) libeay32.dll**
- **(removed) ssleay32.dll**
- **(removed) crtstdll.dll**
- **(removed) esscissl.dll**
- **(removed) sslth.dll**
- **(added) vasslthreads.dll**
-

Unix/Linux OpenSSL and supporting binary changes:

- **(removed) libcrypto.so**
- **(removed) libssl.so**
- **(removed) libcrtfile.so**
- **(removed) libsslthread.so**
- **(added) vasslthreads.so**

Action required

When acquiring OpenSSL, ensure the version is 1.0.0 or above. Make sure to use the 32 bit version.

Download and copy the OpenSSL shared libraries into VA Smalltalk Binary Directory (i.e. same locations as the abt executable).

To minimize abt.ini adjustments, ensure the names of the OpenSSL shared libraries are;

- (Windows) libeay32.dll and ssleay32.dll
- (Unix/Linux) libcrypto.so and libssl.so

In the abt.ini file, ensure the following 3 PlatformLibrary Name Mappings match the following on Windows:

- *CRYPTO_LIB=libeay32*
- *SSL_LIB=ssleay32*
- *THREAD_LIB=vasslthreads*
-

On Unix/Linux, they should be:

- *CRYPTO_LIB=libcrypto*
- *SSL_LIB=libssl*
- *THREAD_LIB=vasslthreads*

Migrating from Version 8.6.2

This chapter addresses concerns for users of VA Smalltalk V8.6.2 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Changed representation of socket address in OSSocketAddrInet and SciSocketAddress
- Cryptography methods modified to support OpenSSL 1.1

Changed representation of socket address in OSSocketAddrInet and SciSocketAddress

Reason for change

Adding support for IPv6 in *SocketCommunicationsInterface* required changing the existing IPv4 socket address representation (an integer in host order) to one compatible (polymorphic) with the IPv6 socket address representation.

Change

OSSocketAddrInet now holds its socket address as a 4-byte ByteArray making it polymorphic with *OSSocketAddrInet6* which holds its socket address as a 16-byte ByteArray. *SciSocketAddress* (since it can hold either an IPv4 or IPv6 socket address) also holds its socket address as a ByteArray.

To conform to the new address representation, the socket constant INADDRANY is changed from 0 to #[0 0 0 0] and the socket constant INADDRNONE is changed from 4294967295 to #[255 255 255 255].

Action required

Find all locations sending *SciSocketAddress>>#address* or *OSSocketAddrInet>>#addr*. These methods now return a 4-byte ByteArray for IPv4 addresses or a 16-byte ByteArray for IPv6 addresses. If you need the integer form of the IPv4 address, you can get it using *aByteArray ntoh32At: 0*.

Find all references to INADDRANY and INADDRNONE. Verify that the usage is valid for addresses represented as ByteArrays.

Cryptography methods modified to support OpenSSL 1.1

Reason for change

OpenSSL 1.1 was a major revamp of the codebase that introduced a lot of breakage in their APIs. We have been able to hide most of this, but there were a few methods that exposed internal representation from OpenSSL which was no longer valid and had to change.

Change

The following methods have been removed:

OSSslCipherCtx>>contextFlags

OSSslCipherCtx>>padding

The following method now answers an <Integer> instead of an <OSSslASN1Integer>

OSSslX509>>version

Action required

After loading the cryptography feature (*ST: Cryptographic Support*), make the appropriate adjustments to your methods that reference any that are described in the section above.

Migrating from Version 8.6.3

This chapter addresses concerns for users of VA Smalltalk V8.6.3 who are migrating to the current release of VA Smalltalk.

Before you beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- PlatformFunction and EsEntryPoint parameter/return type accuracy requirement changed
- PlatformFunction and EsEntryPoint type #handle should not be used
- Restrictions on nested use of #lockThreadIn:

PlatformFunction and EsEntryPoint parameter/ return type accuracy requirement changed

Reason for change

The new virtual machines introduced in V9.0 use a 3rd party library (libffi) to help implement the FFI (Foreign Function Interface) capability of VA Smalltalk. This is the low-level machinery that allows VA Smalltalk to make calls to other languages like C.

In previous versions of the product, all values were extended to 32-bits when being passed to, or returned from, an external language function (except 64-bit types which were passed as 64-bits).

As an example, consider the following C Function.

void doSomething(void *data)

The first parameter type of the associated PlatformFunction should be something like #pointer, since data is of type (void *). However, previous versions of VA Smalltalk would allow you to specify that type as #uint8, #uint16, #uint32. It wouldn't really matter (except 64-bit quantities) because it would just extend it to 32-bits anyways.

While its confusing when the binding type specification does not match the external function specification, the actual reason for this change is simply because this is not how libffi works. It expects a higher degree of type accuracy. The result of specifying #uint8 against the example above would be that the pointer gets truncated to an 8-bit quantity when preparing for the call to the C function.

Action required

Check your existing PlatformFunction and EsEntryPoint usages and ensure that the parameter and return types conform to what is specified by the external function declaration. Pay particular attention to the size of the types to ensure that truncation will not occur.

PlatformFunction and EsEntryPoint type #handle should not be used

Reason for change

The new virtual machines released with V 9.0 introduced a regression related to FFI type #handle. This has been fixed in the virtual machine released with V 9.2.

Action required

For V 9.0 and V 9.1 the workaround is to replace #handle type with #pointer.

Restrictions on nested use of #lockThreadIn:

Reason for change

The restriction is that you can't make another SCI call within the exception handler of the SstTimeout (or any other exception handler invoked while a lockThreadIn: is in process).

This is because, in 8.6.3 thread, management SCI code that was in the essci40.dll was moved into the image.

Action required

No workaround is available.

Migrating from Version 9.0

This chapter addresses concerns for users of VA Smalltalk V9.0 who are migrating to the current release of VA Smalltalk.

Before beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- AbtWaitApp changes
- EsCompressionStreamsApp changes
- EswStreamExtensions changes
- Async Queue Overflow default policy change
- Character conversion default policy change on Unix
- Stream nextLine changes

AbtWaitApp changes

Reason for change

AbtWaitApp was broken on Unix platforms and the codebase needed modernization and improvements.

Action required

The AbtWaitApp subapplications have been **removed** since the OS platform-specific wait functionality is implemented in the virtual machine. This made these subapps unnecessary. Unless you are referring to the subapp names explicitly, there should be no action required. But it's a structure change worth mentioning.

Old API	**New API**
The method AbtConditionalWait>>#perform has been **deprecated**.	Use AbtConditionalWait>>waitAndReturn instead
The method AbtConditionalWait>>setDefaultWaitValues has been **removed**.	These default values are already set in the class side creational methods. @see AbtConditionalWait>>until:ifTimeoutDo:

EsCompressionStreamsApp changes

Reason for change

The EsCompressionStreamsApp was significantly overhauled and improved in 9.1, both in functionality and uniformity of the API. In order to achieve these improvements and make compression streams pluggable with other streams within VA Smalltalk, some API changes had to be made. For most simple usages of these streams, there will be very little change that the user will have to make.

Action required

ZipWriteStream

Before 9.1, a ZipWriteStream was a simple builder objects for a filesystem based MZZipArchive. A 'nextPut' type of API was used to add file paths to the zip file.

The new ZipWriteStream will actually write a valid zip archive to an underlying stream which it decorates. The stream it decorates could be memory, file or socket based. The 'nextPut' API has the more traditional meaning of adding byte/char content to the stream.

If all you need is the original behavior of adding string path names to an existing zip file archive on disk… it is easiest just to follow what the ZipWriteStream did internally.

```
| archive |

"From ZipWriteStream class>>on:"

archive := MZZipArchive basicNew filePath: aFilename; yourself.

"From ZipWriteStream>>nextPutAll:"

archive zipFiles: filenames

"From ZipWriteStream>>close"

archive close.
```

The equivalent of this in the streaming fashion is now:

```
| archive |

archive := ZipWriteStream on: (CfsWriteFileStream openEmpty: aFilename).

filenames do: [:filename |

   archive

        nextPutEntry: (ZipEntry named: filename);
        nextPutAll: 'Content of ', filename].
```

```
archive close.
```

ZipReadStream

ZipReadStreams have been changed in a similar way. Before 9.1, a ZipReadStream was a simple wrapper around an MZZipArchive. It was mostly used to facilitate unzipping file paths but did not have a streaming interface.

The new ZipReadStream will stream across zip entries from a zip archive that might be in memory, on the filesystem, or from a socket. It offers the full streaming API.

If all you need is the original behavior of opening up a filesystem path and unzipping file paths to some directory…this was accomplished by

```
| archive |


"From ZipReadStream class>>on:"

archive := MZZipArchive basicNew openRead: aFilename; yourself.

"From ZipReadStream>>contentsTo:"

archive contentsTo: aPath

"From ZipReadStream>>close"

archive close.
```

The equivalent of this in the new streaming fashion is now:

```
| archive |


archive := ZipReadStream on: (CfsReadFileStream on: aFilename).

[archive nextEntry notNil] whileTrue: [:entry | | out |

     out := CfsWriteFileStream openEmpty: archive entry name.

     out nextPutAll: archive upToEnd; close].
archive close.
```

Deflate/Gzip Streams

As of 9.1, all new compression streams have a unified API and one that should be familiar. The old versions of these streams used APIs like #read and #write. Additionally, #next would answer -1 as a signal for end of stream. In general, it was not possible to wrap other streams whose complete size was not known. This means sockets streams could not be used.

Now, their API is what you would expect from a stream…#next, #nextPut:, #nextPutAll:. #next will throw an exception if there is no more input, but #tryNext will answer nil instead. The API is a superset of all streams in the standard VAST system. These streams are meant to wrap any other stream in the system, which means you should expect to work with them just like any other stream in the system.

It's not encouraged to use the old API, but for clarity their translations will be given below.

Old API	Translation
InflateReadStream>>available	Removed…use atEnd for determining if done instead of a byte count
InflateReadStream>>count	Removed…not used and can't be determined in true streams.
InflateReadStream>>length	Removed…not used and can't be determined in true streams.
InflateReadStream>> read: aByteArray	Use #tryNext: aByteArray
InflateReadStream>>read:atOffset:length:	Use #tryNext:into:startingAt:
InflateReadStream>>next	Use #tryNext but test for nil instead of -1
DeflateWriteStream>>write:atOffset:length	Use #nextPutAll:

EswStreamExtensions changes

Reason for change

Highly reusable Stream method #next:into:startingAt: was moved into base code and out of a swapper related application.

Action required

EswStreamExtensions and EswStreamExtensionsCFS were removed from the product. The extension methods which they defined were pushed to their prerequisites (Kernel and CommonFileSystem).

If you have an application that defines EswStreamExtension as a prerequisite, be sure to remove that prerequisite from your application. Since the methods are now available in Kernel, no other action is required.

If you have an application that defines EswStreamExtensionsCFS as a prerequisite, be sure to replace that prerequisite with CommonFileSystem which is now where those extension methods are defined.

Async Queue Overflow default policy change

The async queue default policy has been to trigger an async overflow error when the async queue overflows. The new default policy is to ignore async queue overflows.

Reason for change

The async queue plays an important role in how UI events are processed on Unix. Sometimes so many events come in at once that the async queue runs out of space and can not add more entries. This results in the runtime error “Async Queue has overflowed” that displays a debugger. Generally, developers simply close the debugger and continue with what they were doing.

The async queue was undersized, which we have changed for 9.1. But additionally, we added a new default policy that async overflows are ignored by default.

While this is the new default policy, it is customizable and the section below describes how to restore the old behavior if you are making use of the async queue in your application and dropping events is not an option.

In the future, we will be building a more elastic async queue that can adapt to the situation.

Action required

To restore the old behavior of triggering an async overflow error, execute the following code on startup of your application.

```
Process ignoreAsyncQueueOverrun: false
```

Character conversion default policy change on Unix

Reason for change

A lot of work was done in v9.1 to unify how character conversion policies work across all platforms. A new policy object (AbtCodePageConversionPolicy) was added to allow the user to customize conversion rules such as whether to skip the UTF-8 BOM or not.

As part of this work, Unix’s default policy was changed to transliterate mode, which means it will try and convert characters in the origin encoding to the closest possible matching character in the target encoding. This unifies it with the policy that existed on Windows.

Before 9.1, the default policy on Unix was strict meaning an error was thrown if an invalid character sequence was detected during conversion.

Action required

To restore the old behavior of strict conversion, change the system wide policy to strict by executing the following code below.

```
AbtCodePageConversionPolicy systemWidePolicy beStrictMode
```

Stream nextLine changes

Reason for change

As of VA Smalltalk 9.1, a stream auto-detects the line delimiter when using the *#nextLine* method. Prior to this behavior, the stream would use the platform line delimiter which made it difficult to create multiline string content on one platform that could be seamlessly read on another with nextLine.

Action required

No action is needed if the auto-detected line delimiter suits your stream, which happens in the majority of applictions.

If the stream's lines are separated using some other line delimiter than the outo-detected one, you can set the delimiter for the stream instance to any arbitrary string by sending the *#lineDelimiter:* message to that stream. A string representing the stream's current line delimiter can be obtained by sending the *#lineDelimiter* message to a stream.

Customizing the line delimiter makes it possible to enforce a specific line delimiter, such as always writing out the Windows line delimiter default, *CrLf*, for certain streams, even for socket protocols, even though the application might be running on Linux. When a given line delimiter is specified by the user, then the stream does not auto-detect anymore and falls back to the pre-version 9.1 behavior that looks only for the specified line delimiter.

Migrating from Version 9.1

This chapter addresses concerns for users of VA Smalltalk V9.1 who are migrating to the current release of VA Smalltalk.

Before beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- EmAttachments changes
- SstImap4Communications changes
- SstSMTPCommunications changes
- NeoJSON WriteStream extension changes
- Manifest file changes some GUI appearance on Windows
- General database changes related storing handles
- Server Runtime's (SRT's) are now self contained
- Deprecate AbtConditionalWait>>#setDefaultWaitValues

EmAttachments changes

The EmAttachments extension to the method String>>#asPath has changed. The method has been renamed to #asEmPath.

Reason for change

The extended method String>>#asPath was overriding the super class implementation of #asPath in EsString from CommonFileSystem which would answer a CfsPath. However, String>#asPath as exteded in EmAttachments would answer an EmPath. Therefore, if EmAttachments was loaded, then sending a string #asPath would answer an instance of EmPath but if EmAttachments was not loaded the same message send would answer an instance of CfsPath. EmPath and CfsPath do not share the exact same API.

Action required

If you had EmAttachments loaded (quite unlikely) and you were using String >> #asPath, then change that to send #asEmPath.

SstImap4Communications changes

SstImap4Communications has been completely revised to be one of the most comprehensive IMAP clients available with over 20 RFCs implemented including popular features such as SSL/TLS, Compression, IDLE (aka Server Push) and more.

Reason for change

The IMAP Support in VAST was minimal and didn't have basic capabilities required to login to modern IMAP servers in a secure way. Network protocols in VAST are becoming more standardized with frameworks and IMAP has been radically improved to meet the new framework standards

Action required

There is not a direct mapping from the old version to the new version. However, the new *SstImap4Client* is very easy to use and there are examples in the *SstImap4ClientExample* class from the SstEmailExamples application that should easily cover any use case the old SstImap4Communications provided.

SstSMTPCommunications changes

SMTP Capability in VAST has been completely revised and modernized. As a result, the legacy application, *SstSMTPCommunications*, has been renamed to *SstSMTPCommunicationsObsolete* and all classes in this application have been renamed slightly to avoid name collisions with the new *SstSmtpCommunications*

Reason for change

The SMTP Support in VAST didn't have basic capabilities required to login to modern SMTP servers in a secure way. Network protocols in VAST are becoming more standardized with frameworks and SMTP has been radically improved to meet the new framework standards.

Action required

The recommended action is to transition to the new *SstSmtpCommunications* and use the *SstSmtpClient* to send emails. There are plenty of example of how to use this in *SstSmtpClientExample* class from the *SstEmailExamples* application.

However, if you are unable to transition just yet, you can load the configuration map called Server Smalltalk – Mail Support Obsolete and you will get the old version of SMTP with classes slightly renamed.

Change all references in your code to the new class names.

NeoJSON WriteStream extension changes

Reason for change

NeoJSON was adding the extension method #<< to *WriteStream*. This protocol is not very common in VA Smalltalk and we prefer to use other messages more polymorphic with other streams.

Action required

Change senders of *WriteStream* >> #<< to use *#nextPutAll:* and *#print:*

Manifest file changes some GUI appearance on Windows

Since version 7.5, a manifest file (abt.exe.manifest) has been shipped and included by default with the product. The inclusion of such a manifest file defers decisions about some GUI appearance elements to Windows. The manifest file (abt.exe.manifest) causes the associated application (abt.exe) to use Windows Common Controls version 6.

It is possible for a user application to be delivered without a manifest. In this case, some Windows User Interface customization may be needed to get the User Interface to look and feel as intended.

Reason for change

The introduction of HiDPI relies on specifying certain assemblies, especially the use of version 6 of Windows common controls.

This also affects the ‘VA: Reports’ feature. In this case, the symptom is grossly large icons after invoking ‘AbtReportPrinter default’.

Action required

If you use the manifest file with your application, no action is required.

If you choose not to use a manifest file, either disable HiDPI or add a customized manifest with just the portion of the manifest which the HiDPI needs.

How do I disable HiDPI?

There are two parts to disabling HiDPI:

In the Smalltalk image, execute `DPIScalerImpl activate` and save the image.

The other step is to either remove the manifest or remove those lines of the manifest that concern HiDPI. Those lines are the *compatibility* and *application "windowsSettings"* portions. As of version 9.2, that portion appears below.

```
  <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
    <application>
        <!-- Windows 10 -->
        <supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}"/>
        <!-- Windows 8.1 -->
        <supportedOS Id="{1f676c76-80e1-4239-95bb-83d0f6d0da78}"/>
        <!-- Windows 8 -->
        <supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>
        <!-- Windows 7 -->
        <supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/>
        <!-- Windows Vista -->
        <supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"/>
    </application>
  </compatibility>
  <application>
    <windowsSettings>
      <dpiAwareness xmlns="http://schemas.microsoft.com/SMI/2016/
WindowsSettings">PerMonitor</dpiAwareness>
      <dpiAware xmlns="http://schemas.microsoft.com/SMI/2005/
WindowsSettings">True/PM</dpiAware>
    </windowsSettings>
  </application>
```

The core is

```
<dpiAwareness
xmlns="http://schemas.microsoft.com/SMI/2016/WindowsSettings">
PerMonitor</dpiAwareness>
```

and

```
<dpiAware
xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">
True/PM</dpiAware>
```

General database changes related storing handles

The database code base no longer holds on to OS memory for storage of various handles related to the database interfaces calls.

Reason for change

When the database code was adapted to the new 32bit/64bit VM and its new memory management protocol, we decided to allocate and manage the memory used for the various database handles directly. We decided to hold on to this memory and reference the handles using the pointerAt: protocol. Upon reflection, we feel these particular changes were a bit confusing and awkward to work with. While the database code is still allocating and managing its OS memory usage, we have decided to only hold on to the handle data, not the OS memory containing the handle.

Action required

If you are simply a consumer of the database code and have not modified or extend the database layer with your own enhancements, then you will not need to do anything.

If you did extend or modify the base database you may need to make some adjustments. A typical reference to a statement handle (stmthp), error handle pointer (errhp), or environment handle (envhp) might look something like this in 9.1:

- self stmthp pointerAt: 0
- self dbConnection errhp pointerAt: 0
- self databaseManager henv pointerAt: 0

Note that there are other handle objects to lookout for.

In 9.2, the database code is no longer holding on the physical OS memory, so the handles are just integer values. In 9.2, you would reference the these handles like this:

- self stmthp
- self dbConnection errhp
- self databaseManager henv

If you miss updating one these types of handle pointers, you will experience errors indicating the Integer does not understand pointerAt: .

Server Runtime's (SRT's) are now self contained

Windows and Linux Server Runtime distributions provide the redistributable files and structure required to run a headless server application.

On Linux, the server runtime zip, contains a bash script called 'installPrereqs' that installs any dependencies on Linux that you may need, like libc or motif.

Execute that script to install the needed dependencies.

Reason for change

The Server Runtime's are dependent on the default VA Smalltalk installation locations. Users are required to modify the supplied .ini and/or script files to point to the local SRT files before they can be used on a system that does not have a development environment installed locally. The Server Runtime distributions should not require a local development environment to work properly or have to edit .ini or script files to reference the SRT redistributable files and folders.

Action required

The new SRT distributions are set up to execute your packaged runtimes from the root SRT folder. You are required to copy the necessary .ini and/or script files up from the newimage folder to the root folder, renaming them as needed.

HiDPI impacts class side OSWidget level extent methods

With the addition of HiDPI capability, some class side OSWidget level extent methods have been removed..

Reason for change

The following public methods have been moved from class-side to instance-side:

- OSComboBox class >> #arrowExtent
- OSDateTimePicker class >> #arrowExtent
- OSButton class >> #checkExtent
- OSButton class >> #hArrowExtent
- OSButton class >> #vArrowExtent
- OSProgressBar class >> #hBarExtent
- OSScrollBar class >> #hBarExtent
- OSTrackBar class >> #hBarExtent
- OSScrollBar class >> #vBarExtent
- OSTrackBar class >> #vBarExtent

In HiDPI environments the class-side methods of the OSWidget and subclasses do not return the correct extent value. The correct extent is defined by an OSWidget (sub)instance displayed on a particular monitor and root desktop window with its own DPI-scaling.

Action required

Convert usage of the class side methods to the instance side.

HiDPI impacts fonts

With the addition of HiDPI capability non scalable fonts will not have a pleasing appearance. If you have chosen non scalable fonts for your UI components, they may not work well if you choose scaling other than 100%.

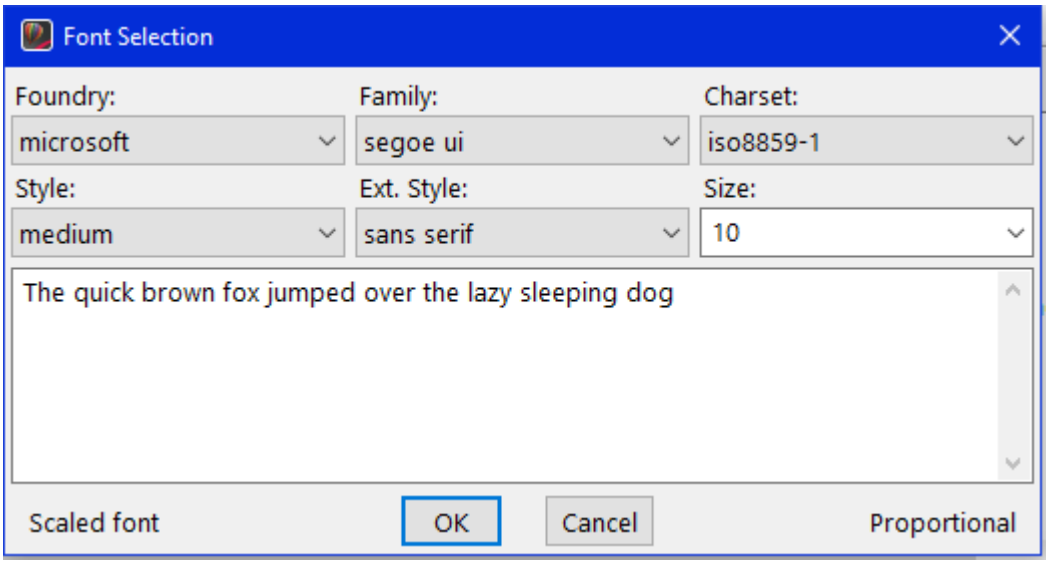
Reason for change

HiDPI requires the use of scalable fonts. Nonscalable fonts are likely to have an unpleasing appearance, especially when a scaling factor is applied.

Action required

No action is needed if you exclusively use scalable fonts.

If you are unhappy with the way your fonts look, convert usage of the non-scalable font to their closest equivalent scalable font. To find a suitable font, use the Composition Editor to change the font property of a User Interface component like an entry field. The Font Dialog looks like this. The ‘Scaled font’ in the lower left of the dialog indicates the font is suitable for usage within a Hi DPI setting.



Deprecate AbtConditionalWait>>#setDefaultWaitValues

AbtConditionalWait>>#setDefaultWaitValues is removed as a method in AbtConditionalWait. Setting of. wait interval, maximum wait, start time is no longer available on the instance side of a conditional wait.

Reason for change

Setting of default values already happens in the class side during instance creation and setting up of the wait.

Action required

If your applications send AbtConditionalWait>>#setDefaultWaitValues you should remove the message.

Migrating from Version 9.2

This chapter addresses concerns for users of VA Smalltalk V9.2 who are migrating to the current release of VA Smalltalk.

Before beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Transcript search results sorted
- HiDPI requires default font change
-

Transcript search results sorted

All the results available from the "Transcript" -> "Tools" -> "Query" are sorted alphabetically. For example the results can be "Open Class Editions", "Unreleased Classes", etc.

Reason for change

The results are sorted in order to allow for more rapid location of the result of interest.

Action required

If you were relying on the report order of the following methods then you would have to update your code to assume they are not sorted alphabetically

- #reportClassesOwnedBy:
- #reportEmptyClassExtensions
- #reportOpenEditions
- #reportUndefinedClasses
- #reportUnreleasedClasses

..

HiDPI requires default font change

Beginning with 9.2 and the introduction of HiDPI for Windows, the default font for User Interface components changed from ‘system-san serif’ to the default system font for Windows 7/10, Microsoft-Segoe UI. This font is a scalable one which blends into Windows and performs well with scaling.

HiDPI applies to Windows only.

Reason for change

HiDPI requires the use of scalable fonts. Nonscalable fonts are likely to have an unpleasing appearance, especially when a scaling factor is applied.

Action required

If you like the new default font, no action is required.

If you want to revert to a default font with characteristics similar to those of ‘system-san serif’, the pre-HiDPI default font which was available in version 9.1 and before, you need to specify this in your INI file.

Find the [CommonGraphics] section of the abt.ini file. You will see two lines:

```
; widgetFontName=pre92SystemEquivalentFontName
```

and

```
widgetFontName=

Uncomment the first and comment out the second.
```

To return to using the new default, Microsoft-Segoe UI, make sure the lines reads as follows.

```
; widgetFontName=pre92SystemEquivalentFontName
widgetFontName=
```

Migrating from Version 9.2.2

This chapter addresses concerns for users of VA Smalltalk V9.2.2 who are migrating to the current release of VA Smalltalk.

Before beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- Non-Inheritable Handles by default on Windows
- Default OpenSSL PlatformLibrary Name Mapping Update
- Exponential notation will now always answer a Float instance

Non-Inheritable Handles by default on Windows

Sockets and CFS File Descriptors will now be specified as non-inheritable on creation.

Reason for change

The new OsProcess framework must inherit handles on Windows to allow VAST to communicate with spawned subprocesses via OS pipes. Sockets or Files closed in VAST may not actually close if their handles have been inherited in a currently running subprocess.

Action required

This is more informational in the unlikely event that a customer is relying on Sockets and File handles being inheritable. If this is the case, it is recommended that you reach out to Instantiations Support to better understand your use case and get you a solution.

Sockets are made non-inheritable in the following method:

SciSocketManager>>platformSocket:type:protocol:

Files are made non-inheritable in the following method:

CfsFileDescriptor class>>extendedOpenCreate:oflag:share:attributes:action:

Default OpenSSL PlatformLibrary Name Mapping Update

The latest supported versions of OpenSSL on Windows use a different naming convention than older versions.

Reason for change

Before VAST 2021, the abt.ini file used libeay32 and ssleay32 as the shared library names for the CRYPTO_LIB and SSL_LIB name mapping entries.

Now, the abt.ini file uses libcrypto and libssl as the shared library names for the CRYPTO_LIB and SSL_LIB name mapping entries to match with the latest supported versions of OpenSSL.

Action required

You may have older versions of OpenSSL installed and in your path. On older versions of VAST, it would have looked for these old names by default. To get back these old names, you must edit the abt.ini file and restore the following name mappings back to:

CRYPTO_LIB=libeay32

SSL_LIB=ssleay32

Exponential notation will now always answer a Float instance

VAST allows the use of Exponential Notation to define numbers, for example "1e2" or "42e10". The Smalltalk parser can use that nomenclature to build number instances. In VAST version 9.2.2 and before those expressions result in Integer instances like 100 (1e2) or 420000000000 (42e10). Starting in VAST 2021 (10.0.0), those expressions answer Float instances: 100.0 (1e2) and 420000000000.0 (42e10).

Reason for change

We believe it's clearer and more correct to always build a Float when you use the Exponential Notation.

Action required

If there are cases where you actually need an Integer instead of a Float you can always send the message #asInteger. For example, "1e2 asInteger".

Migrating from Version 10.0.0

This chapter addresses concerns for users of VAST Platform V10.0.0 who are migrating to the current release of VA Smalltalk/ VAST Platform.

Before beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- EsCodePageUtilitiesApp moved to CodePageSupport
- Object>>#abrAsClass deprecated

EsCodePageUtilitiesApp moved to CodePageSupport

Code page utilities app has been moved into the Kernel as a subapplication and is now visible to the entire system.

Reason for change

Visibility of this application is required by the new UnicodeSupport subapplication in Kernel. To satisfy visibility rules, EsCodePageUtilitiesApp also had to be moved into Kernel as a subapplication.

Action required

The classes/methods inside EsCodePageUtilitiesApp were not renamed and everything should still be visible in your application. However, certain applications in your system may still have EsCodePageUtilitiesApp as a prerequisite.

Be sure to remove EsCodePageUtilitiesApp as a prerequisite from your applications and remove any references to EsCodePageUtilitiesApp that may exist in your packaging instructions.

Object>>#abrAsClass has been deprecated

Object>>#abrAsClass has been marked as deprecated..

Reason for change

The method is the same as Object>>#asClass instead.

Action required

Use Object>>#asClass instead.

Migrating from Version 11.0.0

This chapter addresses concerns for users of VAST Platform V11.0.0 who are migrating to the current release of VAST Platform.

Before beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- "Rectangle" key renamed to "RectangleFunction" in #_PRAGMA_PlatformFunctions

"Rectangle" key renamed to "RectangleFunction" in #_PRAGMA_PlatformFunctions

The following key:

```
(name: Rectangle isConstant: true valueExpression: 'PlatformFunction
fromArray: #(''C'' ''Rectangle'' nil ''gdi32.dll'' #(#pointer #int32 #int32
#int32 #int32) #bool)')
```

was changed to:

```
(name: RectangleFunction isConstant: true valueExpression: 'PlatformFunction
fromArray: #(''C'' ''Rectangle'' nil ''gdi32.dll'' #(#pointer #int32 #int32
#int32 #int32) #bool)')
```

in the method WindowsPlatformFramework class >> _PRAGMA_PlatformFunctions.

Reason for change

If you have a class that uses the pool dictionary PlatformFunctions and you want to write a method that uses the "Rectangle" global, that global will resolve to the PlatformFunction instead of the class Rectangle. We believe this should not be the case and therefore the rename of the key.

Action required

If you have code using the "Rectangle" key of PlatformFunctions, you have to change it to "RectangleFunction"

.

Migrating from Version 11.0.1

This chapter addresses concerns for users of VAST Platform V11.0.1 who are migrating to the current release of VAST Platform.

Before beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- #sstBase64Encoded now uses Base64Coder instead of Base64CoderMime
- On Linux, glibc function ftime() usage in VM has been removed
- LDAP search parameter handling changed

#sstBase64Encoded now uses Base64Coder instead of Base64CoderMime

The method EsString >> sstBase64Encoded was delegating to Base64CoderMime to do the Base 64 encoding. The change is now to delegate to Base64Coder instead.

Reason for change

The method/selector #sstBase64Encoded is too generic to be tight to Base 64 MIME encoding. Such method should be doing instead a regular Base 64 encoding.

Action required

If you really need a Base 64 MIME encoding, then instead of using #sstBase64Encoded, just use Base64CoderMime directly. For example, doing: `Base64CoderMime current encode: 'example'`.

On Linux, glibc function ftime() usage in VM has been removed

The virtual machine primitive VMprTimeClassMillisecondClockValue calls to ftime() have been replaced by calls to clock_gettime().

Reason for change

The Smalltalk expression `Time millisecondClockValue`, which uses the Virtual machine primitive `VMprTimeClassMillisecondClockValue`, has been returning different answers on Linux and

Windows platforms. Upon research, it was determined that ftime() has been removed from glibc starting with version 2.33. Thus, all calls to ftime() in the VM have been replaced with clock_gettime(). With this change Time millisecondClockValue answers a more accurate value.

Action required

None in general, unless your code somehow relies on or have workarounds to non-accurate millisecondClockValue answered in previous versions of the product.

LDAP search parameter handling changed

Parameters passed to the underlying LDAP library are passed in a different way and many ‘PlatformFunction’'s signatures changed its return type from ‘uint32’ to ‘int32’.

Starting with Version 12.0.0, if there is an error in the LDAP search, subsequent operations in the same search will return an ‘LDAPError’ with the code ‘89’ (‘"LDAP_PARAM_ERROR"‘).

Reason for change

To provide better support of different platform architectures and avoid returning wrong codes when there was an error in the platform function call.

Action required

None in general, unless your code somehow relies on or has workarounds to non-accurate `millisecondClockValue` answered in previous versions of the product.

E.g. In previous versions of the product an empty result would return a ‘LDAP_NO_SUCH_OBJECT (32)’ error object, and trying to retrieve the entries afterwards would return the same error object. Starting with version 12.0.0 , an empty result will return ‘LDAP_PARAM_ERROR (89)’.

Migrating from Version 12.0.0

This chapter addresses concerns for users of VAST Platform V12.0.0 who are migrating to the current release of VAST Platform.

Before beginning the migration steps in this chapter, make sure you have completed the pre-migration steps in Preparing for Migration.

After you complete any needed migration steps mentioned in this chapter, you should continue to the subsequent chapters of this document to review further migration requirements.

The topics covered in this chapter include the following:

- CwTable>>#fullRowSelect has been deprecated
- CwTable>>#editLabels has been deprecated
- Instantiation of readers and writers for NeoCSV and NeoJSON has changed
- SstHttpClient>>#transportScheme: has been deprecated

CwTable>>#fullRowSelect has been deprecated

The method CwTable>>#fullRowSelect has been deprecated.

Reason for change

The method overlaps with CwTable>># selectionPolicy.

Action required

Use CwTable>># selectionPolicy.

CwTable>>#editLabels has been deprecated

The methods CwTable>># editLabels and CwTable>># editLabels: have been deprecated.

Reason for change

The method specified whether the first column of a table is editable; i.e. that a user could dire ctly edit cells in the first column only. The CwTable has been redesigned to allow cells in any column to be edited. This suggests a different name for a different capability. That name is *editable*.

Action required

Use CwTable>># editable and CwTable>># editable:.

Instantiation of readers and writers for NeoCSV and NeoJSON has changed

In VAST 12.0.0 and earlier, it was possible to instantiate a NeoJSONReader / NeoCSVReader or NeoJSONWriter / NeoCSVWriter directly using #new. As of VAST 12.0.1, the recommended way to initiate a reader or writer is using the class-side #on: method, and the #new has been made private.

Reason for change

The method #new provided compatibility with its canonical version, where #initialize is sent to the new instance by default. However, VAST Platform doesn't have that by default.

Action required

Instead of

- NeoJSONReader class>>#new
- NeoCSVReader class>>#new
- NeoJSONWriter class>>#new
- NeoCSVWriter class>>#new

use

- NeoJSONReader class>>#on:
- NeoCSVReader class>>#on:
- NeoJSONWriter class>>#on:
- NeoCSVWriter class>>#on:

for instance creation of NeoCSV / NeoJSON reader or writer.

SstHttpClient>>#transportScheme: has been deprecated

In VAST 12.0.0 and earlier, it was possible to instantiate an SstHttpClient using #basicNew and then then setting the instance's #transportScheme:. As of VAST 12.0.1, #transportScheme: has been made a private method and deprecated, since the transportation scheme is already set by the HTTP configuration in the #initializeForConfiguration: method.

Reason for change

The change allows the developer to instantiate a ready-to-use SstHttpClient.

Action required

Use SstHttpClient class>>#forConfiguration or SstHttpClient: class>>#forTransportScheme: to instantiate an SstHttpClient.

Deprecated Classes and Methods

Over the course of many releases, some code is no longer used because of refactoring, error corrections, or one of a number of other reasons. When this happens, the methods that will be removed in the future are marked as *deprecated* and an alternative method is usually suggested (unless the function of the method is simply no longer needed). This provides notification to users of the methods that the methods should not be used and that they will be removed from the product in a future release.

Note:

Private methods are indicated by *italics*. Since private methods should not be referenced in customer code, these methods will likely have a shorter lifespan in the deprecated state.

Original Method	Replacement Method	Deprecated in Version	Removed in Version
AbtBase64Coder class>>current	Base64CoderMime class>>current	8.6.1	12.0.0
AbtBase64Coder class>>reset	Base64CoderMime class>>reset	8.6.1	12.0.0
AbtBase64Coder			12.0.0
AbtConditionalWait>>#perform	AbtConditionalWait>>waitAndReturn	9.1	10.0.0
AbtConditionalWait>> setDefaultWaitValues	default values are already set in the class side creational methods	9.1	10.0.0
AbtSelectorValidator>> isSelectorAlphaNumeric:	Character>> isSmalltalkAlphaNumeric	8.6.2	
AbtSelectorValidator>> isSelectorLetter:	Character>> isSmalltalkLetter	8.6.2	
AbtSelectorValidator>> isSelectorNumber:	Character>> isSmalltalkDigit	8.6.2	
Character>> abtIsIdentifierAlphaNumeric	Character>> isSmalltalkAlphaNumeric	8.6.2	
Character>> abtIsIdentifierLetter	Character>> isSmalltalkLetter	8.6.2	
Color class>> hue:saturation:brightness:	Color class>> h:s:v:	8.6.1	8.6.2
Color>> abtInverse	Color>> inverse	8.6.1	
Color>> caAsInverse	Color>> inverse	8.6.1	
CompiledMethod>> stsDeveloper	CompiledMethod>> developer	8.6	
CwScintillaEditor>> getPasteConvertEndings:	CwScintillaEditor>> getPasteConvertEndings	8.6.1	
CwScintillaEditor>> getStyleBits	n/a	8.6.1	
CwScintillaEditor>> getStyleBitsNeeded	n/a	8.6.1	
CwScintillaEditor>> setStyleBits:	n/a	8.6.1	
CwTable>>#fullRowSelect	CwTable>># selectionPolicy	12.0.1	
CwTable>>editLabels	CwTable>>editable	12.0.1	
DateAndTime class>> year: day: hour: minute: second: millisecond: offset:	DateAndTime class>> year: day: hour: minute: second: millisecond: timeZone:	8.5.2	
DateAndTime class>> year: month: day: hour: minute: second: millisecond: offset:	DateAndTime class>> year: month: day: hour: minute: second: millisecond:timeZone:	8.5.2	
EmClassDefinitionDescription>> hasInvalidCharacter:	String>> isSmalltalkIdentifier not	8.6.2	

EmConfigurationMap>> stsIsLoaded	EmConfigurationMap>> isLoaded	8.5.1	
EmConfigurationMap>> stsIsMissing	EmConfigurationMap>> isMissing	8.5.1	
EmSystemConfiguration>> vaVersion	EmSystemConfiguration>> imageVersion	8.5	
EmSystemConfiguration>> vaVersionDottedId	EmSystemConfiguration>> imageVersionDottedId	8.5	
EsAbstractMethodBrowser>> stsDeveloperString	EtAbstractMethodBrowser>> developerString	8.6	8.6.3
EsIdentitySet	IdentitySet	8.0.2	
EsLogManager class>> error:	EsLogManager class>> logError:	8.6	8.6.3
EsLogManager class>> debug:	EsLogManager class>> logDebug:	8.6	
EsLogManager class>> info:	EsLogManager class>> logInfo:	8.6	8.6.3
EsLogManager class>> warn:	EsLogManager class>> logWarn:	8.6	
EsLogManager class>> debug:object:	EsLogManager class>> logDebug:object	8.6	
EsLogManager class>> error:object:	EsLogManager class>> logError:object:	8.6	
EsLogManager class>> info:object:	EsLogManager class>> logInfo:object:	8.6	
EsLogManager class>> warn:object:	EsLogManager class>> logWarn:object:	8.6	
EsLogManager class>> debug:locationInfo:	EsLogManager class>> logDebug:locationInfo:	8.6	
EsLogManager class>> error:locationInfo:	EsLogManager class>> logError:locationInfo:	8.6	
EsLogManager class>> info:locationInfo:	EsLogManager class>> logInfo:locationInfo:	8.6	
EsLogManager class>> warn:locationInfo:	EsLogManager class>> logWarn:locationInfo:	8.6	
EsLogManager class>> debug: locationInfo: object:	EsLogManager class>> logDebug: locationInfo: object:	8.6	
EsLogManager class>> error: locationInfo: object:	EsLogManager class>> logError: locationInfo: object:	8.6	
EsLogManager class> I nfo: locationInfo: object:	EsLogManager class> >logInfo: locationInfo: object:	8.6	
EsLogManager class> >warn: locationInfo: object:	EsLogManager class>> logWarn: locationInfo: object	8.6	
EsString>> abtIsSmalltalkIdentifier	EsString>> isSmalltalkIdentifier	8.6.2	
EsString>> abtIsValidMethodName	EsString> >isSmalltalkIdentifier	8.6.2	
EtAbstractMethodsBrowser>> stsDeveloperString	EtAbstractMethodsBrowser>> developerString	8.6	
EtWindowSystemStartUp class>> #readConfigurationFile	*EtWindowSystemStartUp class>>* #readConfigurationFilePreStartUp	8.6.2	
ExceptionalEvent>> ancestorOf:	ExceptionalEvent>> handles:	8.0.2	
ExceptionalEventCollection>> ancestorOf:	ExceptionalEventCollection>> handles:	8.0.2	
Object>> abtAsClass	Object>> asClass	8.6.2	11.0.0
Object>>abrAsClass	Object>> asClass	8.6.2	11.0.0
Object>> abtIsProcess	Object>> isProcess	8.6.1	
Object>> instVarAtName:	Object>> instVarNamed:	8.0.2	
Object>> instVarAtName:put:	Object>> instVarNamed:put:	8.0.2	

OpenSSLCryptoLibraryDispatcher>> callOpenSSL_add_all_algorithms:	*OpenSSLCryptoLibraryDispatcher>> callOpenSSL_add_all_algorithms_noconf*	8.6.2	
OpenSSLCryptoLibraryDispatcher>> callPEM_ASN1_readWith: with: with: with: with: with:		8.6.2	
OSCall>> initCommonControls	OSCall> >initCommonControlsEx	8.5	
OSCall>> multiByteToWideChar: dwFlags: lpMultiByteStr: cchMultiByte: lpWideCharStr: cchWideChar:	OSCall>> multiByteToWideChar: dwFlags: lpMultiByteStr: cbMultiByte: lpWideCharStr: cchWideChar:	8.6.1	
OSCall>> setMessageQueue:	n/a	8.6	
OSCall>> wideCharToMultiByte: dwFlags: lpWideCharStr: cchWideChar: lpMultiByteStr: cchMultiByte: lpDefaultChar: lpUsedDefaultChar:	OSCall>> wideCharToMultiByte: dwFlags: lpWideCharStr: cchWideChar: lpMultiByteStr: cbMultiByte: lpDefaultChar: lpUsedDefaultChar:	8.6.1	
OSHostent	OSHostEnt	8.6.2	
OSProtoent	OSProtoEnt	8.6.2	
OSServent	OSServEnt	8.6.2	
OSSslAsn1Integer class>> newWithValue:	OSSslAsn1Integer class>> createNewFromInteger:	8.6.2	
OSSslCtx class>> newForMethod:	*OSSslCtx class>> createNewFromMethod:*	8.6.2	
OSSslRSAPrivateKey	OSSslRSAKey	8.6.2	
OSSslSession class>> newSessionFromSslHandle:	OSSslSSL>> getSession	8.6.2	9.2.2
OSSslSession>> setSessionForSslHandle:	OSSslSession>> setSession:	8.6.2	9.2.2
OSSslSSLMethod class>> #SSLv2	OSSslSSLMethod class> >#SSLv23	8.6.2	
OSSslSSLMethod class>> #SSLv2_client	OSSslSSLMethod class>> #SSLv23_client	8.6.2	
OSSslSSLMethod class> >#SSLv2_server	OSSslSSLMethod class>> #SSLv23_server	8.6.2	
OSSslSSLMethod class>>#SSLv3	OSSslSSLMethod class> >#SSLv23	8.6.2	
OSSslSSLMethod class>>#SSLv3_client	OSSslSSLMethod class>> #SSLv23_client	8.6.2	
OSSslSSLMethod class>>#SSLv3_server	OSSslSSLMethod class>> #SSLv23_server	8.6.2	
OSSslStructure class>>builtInstanceOrErrorFromAddress:	OSSslStructure class>> createSmartReferenceTo:	8.6.2	
OSSslX509 class>>fromDerFile:	OSSslX509 class>> createNewFromFile:	8.6.2	
OSSslX509 class>>fromPemFile:	OSSslX509 class>> createNewFromFile:	8.6.2	
OSSslX509>>getIssuerName	OSSslX509>> issuer oneLine	8.6.2	
OSSslX509>>getNotAfterDateAndTime	OSSslX509>> notAfter asDateAndTime	8.6.2	
OSSslX509>>getNotBeforeDateAndTime	OSSslX509>> notBefore asDateAndTime	8.6.2	
OSSslX509>>getSerialNumber	OSSslX509>> serialNumber asInteger	8.6.2	
OSSslX509>>getSubjectName	OSSslX509>> subject oneLine	8.6.2	
OSSslX509>>setSerialNumber	OSSslX509> >serialNumber:	8.6.2	

OSStructure>>fix	OSStructure>> makeReferenceFixed	8.6.2	
Pragma>>analogousCodeTo:	n/a	8.6.1	
Pragma>> hasLiteral:	n/a	8.6.1	
Pragma>> hasLiteralSuchThat:	n/a	8.6.1	
ScaledDecimal>>asDecimal	ScaledDecimal>> asScaledDecimal	8.5	
ScaledDecinal>> precision:	*ScaledDecimal>> privatePrecision:*	8.5	
ScaledDecimal>> scale:	*ScaledDecimal>> privateScale:*	8.5	
SciSocketManager class>> isCompatible	n/a	8.6.3	
SciSocketManager class>> platformIndentifier	System>> osType	8.6.3	
SciSslSocketConfiguration>> loadRandomBytes:	OSSslRandom class>> loadRandomBytes:	8.6.2	
SciSslSocketConfiguration>> loadRandomBytes: upTo:	OSSslRandom class>> loadRandomBytes: upTo:	8.6.2	
SstBase64Coder class>> new	Base64CoderMime class>> new	8.6.1	12.0.0
SstBase64Coder			12.0.0
SstConnection>>receiveIgnoringResponseContent:	SstConnection>>receive:	11.0.0	
SstHttpClient>>#transportScheme	SstHttpClient class>>#forConfiguration or SstHttpClient class>>#forTransportScheme:	12.0.1	
SstLocalEndpoint>>receiveIgnoringResponseContent:	SstLocalEndpoint>>receive:	11.0.0	
SstTransport>>receiveIgnoringResponseContent:	SstTransport>>receive:	11.0.0	
SstTransport>> receiveNextRequestOn: ignoringResponseContent:	SstTransport>> receiveNextRequestOn:assemblerPolicy:	11.0.0	
StsColorText class>> exiting	n/a	8.6	
StsColorText class>> fontList	StsCodeFontManager>> fontList	8.6	
StsColorText class>> fontList:	StsCodeFontManager>> fontList:	8.6	
StsColorText class>> fontName	StsCodeFontManager>> fontName	8.6	
StsColorText class>> preStartUp	n/a	8.6	
StsColorText class>> updateCodeFontList:	StsCodeFontManager>> updateCodeFontList:	8.6	
StsPowerTools class>> currentColorScheme	StsPowerTools class>> currentEditorTheme	8.6.1	
StsPowerTools class>> currentColorScheme:	StsPowerTools class>> currentEditorTheme:	8.6.1	
StsPowerTools class>> currentColorSchemeMap	StsPowerTools class>> currentEditorThemeMap	8.6.1	
WbApplication class>> isWin32s	n/a	8.6.1	
WbIconManager	EwIconManager hierarchy	8.0.3	
WkIconManager	EwIconManager hierarchy	8.0.3	

Migration Tools

Two tools are included with VA Smalltalk to assist you in migrating from earlier versions of VisualAge Smalltalk or VA Smalltalk to the current version of VA Smalltalk. These tools may be separate executables, workspaces that contain code for you to swipe and execute, or applications that you can load into your image.

While these tools were all developed to assist in migration, you may find some of them useful in other scenarios.

The following tools are described:

- Library Importer
- Export/Import Passive Image Properties

Library Importer Tool

It is often useful to be able to import maps from one development library into another, particularly when migrating

- from VisualAge Smalltalk to VA Smalltalk or
- from one version of VA Smalltalk to another or
- from one version of VA Smalltalk to VAST Platform (VA Smalltalk).

The *importer* program is the tool that provides this function. Both the client and the manager product installation processes install the *importer* program in the `<varoot>\importer` directory.

Windows: The *importer* program is called `importer.exe`.

Linux: The the *importer* program is called `importer`.

The *importer* is most useful when installing a new version of the product where it can be used to update your working development manager with the content of the new version's manager. You typically would not use the *importer* to merge a working development library into the new version's library since the *importer* does not copy everything from the source library; it copies only the **versioned** configuration maps (unless `allVersions=false` is specified, in which case only the **newest versioned** configuration mapswill be copied).

The *importer* accepts the following parameters:

-z.target= accepts the location of the target library – the development library that will contain the imported code. The library can be specified in one of two ways:

libraryPath This will use fileIO to access the library

server::libraryPath This will use EMSRV to access the library

-z.source= accepts the location of the source library – the development library containing the code to be copied. The format is the same as for the target.

-z.sourcedir= accepts a directory path. Code will be copied from of each library (i.e. each file with a DAT extnsion) in the directory.

-**z.allVersions**= accepts 'true' to import all map versions from the source library or libraries, or 'false' to import only the most recent version of each map in the source library or libraries. Default is 'true'.

-z.silent runs with no user interface (no longer used, it will be ignored).

Notes:

1. Either `-z.source` or `-z.sourcedir` must be specified, but never both.
2. With File I/O only one person may be using the development library at any one time or you will get the following error message:
   `Error 65: Open Failed.`
3. The **z.** prefix on parameters is optional as of V8.6.3.

After the tool runs, check for non-zero exit codes. The tool will end with a 0 (zero) exit code if the import was successful. If the import was not successful, the tool will end with a non-zero exit code.

The following are some examples of how to run the tool. All examples copy from a working development library into the new development library, `mgr1201.dat.`

- `importer.exe -source=e:\tmp\myfeat.dat -target=localhost::d:\vamgr\manager\mgr1201.dat`
- `importer.exe "-source=C:\Program Files\Instantiations\VA Smalltalk\9.2.2\manager\mgr-working.dat" "-target =C:\Program Files\Instantiations\VASTPlatform\12.0.0x64\manager\mgr1201.dat"`
- `importer.exe -z.target=zot::d:\vamgr\manager\mgr1201.dat -z.sourcedir=e:\tmp\featureDir -z.allVersions=true`

The shows how to run the tool to copy from a working development library into the new development library, mgr922.dat. (the last version of VA Smalltalk)

- `importer.exe "-source=c:\Program Files\Instantiations\VA Smalltalk\9.2\manager\mgr-working.dat" "-target =c:\Program Files\Instantiations\VA Smalltalk\9.2\manager\mgr922.dat"`

The following examples copies from the new development library, mgr1201.dat, into a working development library.

- `importer.exe "-target=c:\Program Files\Instantiations\VA Smalltalk\12.0\manager\mgr-working.dat" "-source=c:\Program Files\Instantiations\VASTPlatform\12.0.0x64\manager\mgr1201.dat"`

The following examples copies from the new development library, mgr922.dat. (the last version of VA Smalltalk), into a working development library.

- `importer.exe "-target=c:\Program Files\Instantiations\VA Smalltalk \12.0\manager\mgr-working.dat" "-source=c:\Program Files\Instantiations\VA Smalltalk\9.2\manager\ mgr922`.dat"

Logging

An `importer.ini` file is provided in the `<vast>\importer` directory. It contains an `[EmLibraryImporter]` stanza controlling the logging done by the importer. You can set the value of `logFilename=` to empty (blank) to disable logging or to the fully-qualified name of the desired log file to enable logging. The default value is

Linux: `logFilename=temp/importer.log` or

Windows: `logFilename=%tmp%\importer.log`

Export/Import Passive Image Properties

In order to facilitate migration of XD passive images from one version of VA Smalltalk to another (or from one image to another), the *Server Workbench, Base* feature has been extended to allow the transfer of the properties of a passive image via a text file.

The steps required are as follows:

1. In the source (old) development image, write the properties of a named passive image to a file. Use one of the Smalltalk expressions

   ```
   EmImageManager
      writePassiveImagePropertiesFor: 'myPassiveImage'
      inFile: 'myPassiveImage.txt'.

   EmImageManager writePassiveImagePropertiesFor: 'myPassiveImage'
   ```

   The first expression allows you to specify the file in which the properties are recorded. The second expression creates a text file with the same name as the passive image.

2. In the target (new) development image, load the *Server Workbench, Base* feature. You may also need to load other features used in the passive image into the development image, e.g. *Server, Database* and *Server, SST*.
3. In the new development image, create a new passive image using the expression:

   ```
   EmImageManager newImageWithPropertiesFrom: 'myPassiveImage.txt'.
   ```

   This results in an Image Properties dialog with the property values from the text file `myPassiveImage.txt`. If there were any problems discovered in setting the properties, they are recorded in the transcript.

   Click on '**OK**' to create a new passive image with the properties shown in the Image Properties dialog.