VA Smalltalk Migration Guide
Tools, Tips & Techniques for Upgrading From IBM's VisualAge® Smalltalk
Instantiations
# Table of Contents

Preface ........................................................................................................................................................................... 1

VA Smalltalk Migration Guide ........................................................................................................................................ 2

Notices .............................................................................................................................................................................. 3

  Trademarks ................................................................................................................................................................. 3

About this book ............................................................................................................................................................... 5

  Who this book is for ............................................................................................................................................. 5

  What's new? ......................................................................................................................................................... 5

  Conventions used in this book ............................................................................................................................... 5

  Tell us what you think ....................................................................................................................................... 6

Introduction to Migration ................................................................................................................................................. 7

Preparing for Migration .................................................................................................................................................. 8

  Merge applications from the existing library to the new library ......................................................................... 8

  Merge the new library into your existing library ................................................................................................. 8

Migrating from VisualAge Smalltalk V3.0 or earlier ............................................................................................... 9

  Migrating applications from VisualAge Smalltalk Standard .............................................................................. 9

    Generating runtime code ................................................................................................................................... 9

Using the migration tool ............................................................................................................................................... 10

  Running the tool .................................................................................................................................................. 10

Migrating database applications ................................................................................................................................ 10

  Loading database migration support ................................................................................................................... 11

  Database Manager .............................................................................................................................................. 11

    Migrating your application ............................................................................................................................... 11

  Database Access Set Runtime ............................................................................................................................ 12

  Dynamic to Static ............................................................................................................................................... 12

Swapper ......................................................................................................................................................................... 13

  General migration .............................................................................................................................................. 13

  API maintained for compatibility ....................................................................................................................... 13
Dumping, loading, and the Common File System subsystem......................................................... 14
  Example: Unloading using CfsFileDescriptor................................................................. 14
  Example: Loading using CfsFileDescriptor................................................................. 14
Other migration concerns.......................................................................................................................... 15
  Enabling applications from Version 1.0 or 2.0 for packaging........................................ 15
  Promoted features of a Container Details View.............................................................. 15
  Updating connections............................................................................................................. 15
  References to pool dictionary CwConstants, AbtCwConstants, or EwConstants............. 16
  AbtNormalGraphic constant used in nonvisual parts....................................................... 16
  Web Connection................................................................................................................. 17
  Extensions of Decimal class.............................................................................................. 17
  Applications that use OSObject subclasses....................................................................... 17
    Migrating OSObject subclasses..................................................................................... 18
    Migrating code that instantiates OSObject directly....................................................... 18
  View wrappers that use attribute-to-attribute connections........................................... 19
  Migrating the default font for Report Writer reports..................................................... 20
  Packager warning 'No implementors of fixupNlsPool'................................................... 20
  EwIconTree migration may cause walkback.................................................................... 21
    Solution......................................................................................................................... 22
Migrating from Version 4.0, 4.01, or 4.02.................................................................................. 23
  Packager changes.............................................................................................................. 23
    Symbol and class reference adjustment API.............................................................. 23
    Priority for packaging rules......................................................................................... 23
    Problem and rule policies............................................................................................ 24
      Problem policies........................................................................................................... 24
      Rule policies............................................................................................................... 24
  Migrating distributed applications................................................................................... 24
    Distributed Smalltalk infrastructure............................................................................ 25
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object space</td>
<td>25</td>
</tr>
<tr>
<td>Description</td>
<td>25</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>25</td>
</tr>
<tr>
<td>Communications</td>
<td>26</td>
</tr>
<tr>
<td>Description</td>
<td>26</td>
</tr>
<tr>
<td>SST Equivalent</td>
<td>26</td>
</tr>
<tr>
<td>Marshaling</td>
<td>26</td>
</tr>
<tr>
<td>Description</td>
<td>26</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>26</td>
</tr>
<tr>
<td>Logical process</td>
<td>26</td>
</tr>
<tr>
<td>Description</td>
<td>26</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>26</td>
</tr>
<tr>
<td>Activation</td>
<td>27</td>
</tr>
<tr>
<td>Description</td>
<td>27</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>27</td>
</tr>
<tr>
<td>Security</td>
<td>27</td>
</tr>
<tr>
<td>Description</td>
<td>27</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>27</td>
</tr>
<tr>
<td>Distributed development tools</td>
<td>27</td>
</tr>
<tr>
<td>Remote Workspace</td>
<td>27</td>
</tr>
<tr>
<td>Description</td>
<td>27</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>28</td>
</tr>
<tr>
<td>Remote Transcript</td>
<td>28</td>
</tr>
<tr>
<td>Description</td>
<td>28</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>28</td>
</tr>
<tr>
<td>Remote file dialog</td>
<td>28</td>
</tr>
<tr>
<td>Description</td>
<td>28</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>28</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Distributed Inspector</td>
<td>28</td>
</tr>
<tr>
<td>Description</td>
<td>28</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>28</td>
</tr>
<tr>
<td>Distributed Debugger</td>
<td>29</td>
</tr>
<tr>
<td>Description</td>
<td>29</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>29</td>
</tr>
<tr>
<td>Distributed garbage collection</td>
<td>29</td>
</tr>
<tr>
<td>Description</td>
<td>29</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>29</td>
</tr>
<tr>
<td>Distributed load</td>
<td>29</td>
</tr>
<tr>
<td>Description</td>
<td>29</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>30</td>
</tr>
<tr>
<td>Name server</td>
<td>30</td>
</tr>
<tr>
<td>Description</td>
<td>30</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>30</td>
</tr>
<tr>
<td>Event profiling (method calls)</td>
<td>30</td>
</tr>
<tr>
<td>Description</td>
<td>30</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>31</td>
</tr>
<tr>
<td>VA Smalltalk Parts</td>
<td>31</td>
</tr>
<tr>
<td>Description</td>
<td>31</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>31</td>
</tr>
<tr>
<td>Distribution menu</td>
<td>31</td>
</tr>
<tr>
<td>Description</td>
<td>31</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>31</td>
</tr>
<tr>
<td>Distributed runtime APIs</td>
<td>31</td>
</tr>
<tr>
<td>Basic remote object API</td>
<td>32</td>
</tr>
<tr>
<td>Description</td>
<td>32</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>32</td>
</tr>
<tr>
<td>Porting guidelines</td>
<td>32</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>Object copying API</td>
<td>32</td>
</tr>
<tr>
<td>Description</td>
<td>33</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>33</td>
</tr>
<tr>
<td>Porting guidelines</td>
<td>33</td>
</tr>
<tr>
<td>Parallel processing API</td>
<td>34</td>
</tr>
<tr>
<td>Description</td>
<td>34</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>34</td>
</tr>
<tr>
<td>Porting guidelines</td>
<td>34</td>
</tr>
<tr>
<td>Name server</td>
<td>34</td>
</tr>
<tr>
<td>Description</td>
<td>34</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>35</td>
</tr>
<tr>
<td>Porting guidelines</td>
<td>35</td>
</tr>
<tr>
<td>Event callbacks (connect/disconnect)</td>
<td>36</td>
</tr>
<tr>
<td>Description</td>
<td>36</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>36</td>
</tr>
<tr>
<td>Porting guidelines</td>
<td>36</td>
</tr>
<tr>
<td>Distributed exceptions</td>
<td>36</td>
</tr>
<tr>
<td>Description</td>
<td>36</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>37</td>
</tr>
<tr>
<td>Porting guidelines</td>
<td>37</td>
</tr>
<tr>
<td>Object space management</td>
<td>37</td>
</tr>
<tr>
<td>Description</td>
<td>37</td>
</tr>
<tr>
<td>SST equivalent</td>
<td>38</td>
</tr>
<tr>
<td>Porting guidelines</td>
<td>38</td>
</tr>
<tr>
<td>Distributed runtime APIs</td>
<td>39</td>
</tr>
<tr>
<td>DS-API</td>
<td>39</td>
</tr>
<tr>
<td>DT-API</td>
<td>40</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Other migration concerns</td>
<td>40</td>
</tr>
<tr>
<td>Using VA Smalltalk features in server applications</td>
<td>40</td>
</tr>
<tr>
<td>Runtime code for nonvisual parts in server applications</td>
<td>41</td>
</tr>
<tr>
<td>Headless runtime applications</td>
<td>41</td>
</tr>
<tr>
<td>Signalling events with arguments</td>
<td>41</td>
</tr>
<tr>
<td>Ending server applications</td>
<td>42</td>
</tr>
<tr>
<td>MVS applications that reference subsystem type 'TM'</td>
<td>42</td>
</tr>
<tr>
<td>Smalltalk error message in a CICS workstation environment</td>
<td>43</td>
</tr>
<tr>
<td>Search path for multiple images of Smalltalk using CICS classes</td>
<td>43</td>
</tr>
<tr>
<td>Snapshot files</td>
<td>43</td>
</tr>
<tr>
<td>Migrating from Version 4.5</td>
<td>44</td>
</tr>
<tr>
<td>Formatted Text Field part</td>
<td>44</td>
</tr>
<tr>
<td>Enhanced shared library support on AIX</td>
<td>45</td>
</tr>
<tr>
<td>Server applications using .mpr files</td>
<td>45</td>
</tr>
<tr>
<td>Migrating from Version 5.0</td>
<td>46</td>
</tr>
<tr>
<td>Hiding Java packages and projects in browsers</td>
<td>46</td>
</tr>
<tr>
<td>Providing a customized splash screen</td>
<td>46</td>
</tr>
<tr>
<td>Migrating control files</td>
<td>47</td>
</tr>
<tr>
<td>Importing feature code</td>
<td>49</td>
</tr>
<tr>
<td>Migrating method mappings to JDK 1.2</td>
<td>49</td>
</tr>
<tr>
<td>Changes in Server Workbench</td>
<td>50</td>
</tr>
<tr>
<td>Current code page support is inconsistent on supported platforms</td>
<td>50</td>
</tr>
<tr>
<td>AbtConnectionSpec &gt;&gt;#targetCodePage: expects a String argument</td>
<td>50</td>
</tr>
<tr>
<td>SST HTTP Servlet API changed</td>
<td>50</td>
</tr>
<tr>
<td>POSIX Migration on OS/390</td>
<td>51</td>
</tr>
<tr>
<td>Behavior changes for Application Builder parts</td>
<td>51</td>
</tr>
<tr>
<td>Migrating from Version 5.5</td>
<td>53</td>
</tr>
<tr>
<td>Elimination of required map links in configuration maps</td>
<td>53</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Low-level configuration map reorganization</td>
<td>53</td>
</tr>
<tr>
<td>Time class shape changed</td>
<td>54</td>
</tr>
<tr>
<td>Default XML serialization methods added for Date and Time.</td>
<td>54</td>
</tr>
<tr>
<td>Changed behavior of millisecondClockValue on OS/390.</td>
<td>54</td>
</tr>
<tr>
<td>Behavior change in handling negative system time.</td>
<td>54</td>
</tr>
<tr>
<td>Behavior change in creating FileStreams.</td>
<td>55</td>
</tr>
<tr>
<td>Packager 'do not reduce' rule behavior changed.</td>
<td>55</td>
</tr>
<tr>
<td>EMSRV startup parameter changes.</td>
<td>56</td>
</tr>
<tr>
<td>Migrating from Version 6.0</td>
<td>57</td>
</tr>
<tr>
<td>AbtPortableNotebookPageView inconsistent behavior</td>
<td>57</td>
</tr>
<tr>
<td>ScaledDecimal&gt;&gt;#hash changed</td>
<td>58</td>
</tr>
<tr>
<td>Subtle change to Compiler</td>
<td>58</td>
</tr>
<tr>
<td>Using ‘Class Variable Initializer’ packaging rules for IC packaging.</td>
<td>59</td>
</tr>
<tr>
<td>Removed unused method #abtXmlPrintOn:namespaces:</td>
<td>59</td>
</tr>
<tr>
<td>Suppress serialization of nil attribute values when attribute mapping specifies Required=&quot;false&quot;</td>
<td>59</td>
</tr>
<tr>
<td>Migrating from Version 7.0</td>
<td>60</td>
</tr>
<tr>
<td>Windows theme support interferes with some testing tools.</td>
<td>60</td>
</tr>
<tr>
<td>Change</td>
<td>60</td>
</tr>
<tr>
<td>Action Required</td>
<td>60</td>
</tr>
<tr>
<td>SST HTTP Server access log.</td>
<td>60</td>
</tr>
<tr>
<td>Migrating from Version 7.5</td>
<td>61</td>
</tr>
<tr>
<td>Modifiable splash screen on Windows</td>
<td>61</td>
</tr>
<tr>
<td>Consolidate EtToolsVendorExtensionsApp and VisualAgeVendorExtensionsApp</td>
<td>61</td>
</tr>
<tr>
<td>Server Smalltalk (SST) feature definition changes.</td>
<td>61</td>
</tr>
<tr>
<td>Merge ENVY/Image Controls apps into other maps.</td>
<td>62</td>
</tr>
<tr>
<td>Change</td>
<td>63</td>
</tr>
<tr>
<td>Action required</td>
<td>63</td>
</tr>
<tr>
<td>Changes to the AbtTimestamp class.</td>
<td>63</td>
</tr>
<tr>
<td>Reason for change</td>
<td>Change</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
</tr>
<tr>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>New testing and conversion CLDT API methods</td>
<td>64</td>
</tr>
<tr>
<td>Manifest file changes some GUI appearance on Windows</td>
<td>64</td>
</tr>
<tr>
<td>Merge AbtCwControlsEdit and AbtCwControlsRun apps into other maps</td>
<td>64</td>
</tr>
<tr>
<td>Change to SstHttpServletRequest&gt;#getContentType</td>
<td>65</td>
</tr>
<tr>
<td>Migrating from Version 8.0</td>
<td>66</td>
</tr>
<tr>
<td>Changes to the UNIXProcess class</td>
<td>66</td>
</tr>
<tr>
<td>Change to default RMI stubProtocolVersion</td>
<td>66</td>
</tr>
<tr>
<td>Change to exception handling</td>
<td>67</td>
</tr>
<tr>
<td>Change to Dictionary&gt;#copy behavior</td>
<td>67</td>
</tr>
</tbody>
</table>
Action required...........................................................................................................................................67

AbtWaitApp moved from IBM Smalltalk, ALL - UI Run to IBM Smalltalk, ALL - Headless configuration map..................................................................................................................................68
  Reason for the change..........................................................................................................................68
  Change..................................................................................................................................................68
  Action required................................................................................................................................68

New testing CLDT API methods.............................................................................................................68
  Object>>#initialize method added........................................................................................................68
    Reason for change.............................................................................................................................69
    Change..............................................................................................................................................69
    Action required................................................................................................................................69

Migrating from Version 8.0.1.....................................................................................................................70
  Grease Core and Grease Core Tests maps are obsolete.........................................................................70
    Reason for change.............................................................................................................................70
    Change..............................................................................................................................................70
    Action required................................................................................................................................70

EsString>>#subStrings: changed to be ANSI-compliant and portable......................................................70
  Reason for the change..........................................................................................................................70
  Change..................................................................................................................................................71
  Action required....................................................................................................................................71

Random class and methods referring to it have been removed.................................................................72
  Reason for change...............................................................................................................................72
  Change..................................................................................................................................................72
  Action required....................................................................................................................................72

SequenceableCollection>>#beginsWith: and #endsWith: have been replaced........................................72
  Reason for change...............................................................................................................................72
  Change..................................................................................................................................................72
  Action required....................................................................................................................................72
Duration>>#milliseconds changed to be portable................................................................. 73
  Reason for change........................................................................................................ 73
  Change............................................................................................................................ 73
  Action required............................................................................................................ 73
IdentitySet class added as replacement for EsIdentitySet............................................... 73
  Reason for change........................................................................................................ 73
  Change............................................................................................................................ 73
  Action required............................................................................................................ 73
Color class promoted from WbKernel to Kernel.............................................................. 73
  Reason for change........................................................................................................ 73
  Change............................................................................................................................ 73
  Action required............................................................................................................ 73
SortedCollection private methods #sorted and #sorted: renamed................................. 74
  Reason for change........................................................................................................ 74
  Change............................................................................................................................ 74
  Action required............................................................................................................ 74
Migrating from Version 8.0.2.......................................................................................... 75
Object>>#value added to CLDT......................................................................................... 75
  Reason for change........................................................................................................ 75
  Change............................................................................................................................ 75
  Action required............................................................................................................ 75
Icon management classes have been reworked............................................................ 75
  Reason for change........................................................................................................ 75
  Change............................................................................................................................ 75
  Action required............................................................................................................ 76
Mastering ENVY/Developer Configuration Expression Prompter configuration map removed........ 76
  Reason for change........................................................................................................ 76
  Change............................................................................................................................ 77
Table of Contents

Action required....................................................................................................................................77
Changes to ScaledDecimal>>#printOn: and #printString........................................................................ 77
Reason for change...............................................................................................................................77
Change................................................................................................................................................. 77
Action required....................................................................................................................................77
Changes to TrailBlazer Browsers..............................................................................................................77
Reason for change...............................................................................................................................77
Change................................................................................................................................................. 77
Action required....................................................................................................................................78
Migrating from Version 8.0.3......................................................................................................................... 79
Consolidation of EsbBenchmarksES subapplication................................................................................ 79
Reason for change...............................................................................................................................79
Change................................................................................................................................................. 79
Action required....................................................................................................................................79
Change Float and Scaled Decimal 32-bit hash for better distribution......................................................79
Reason for change...............................................................................................................................79
Change................................................................................................................................................. 79
Action required....................................................................................................................................79
INI file changes introduced by Preference Settings Framework.............................................................. 80
Reason for change...............................................................................................................................80
Change................................................................................................................................................. 80
Action required....................................................................................................................................80
Change to CgScreen class>>bitmapPath and Locale class>>nlsPath.......................................................81
Reason for change...............................................................................................................................81
Change................................................................................................................................................. 81
Action required....................................................................................................................................81
Migrating from Version 8.5............................................................................................................................ 82
Changed specification of ODBC dll or shared object in .INI file............................................................ 82
Reason for change............................................................................................................................... 82
Change................................................................................................................................................. 82
Action required.................................................................................................................................... 82
Changed PlatformConstants::Error to PlatformConstants::Errorregion for Windows......................... 82
Reason for change............................................................................................................................... 82
Change................................................................................................................................................. 83
Action required.................................................................................................................................... 83
Added 2 Parameters to the socketAppender Setting in the ini File.......................................................... 83
Reason for change............................................................................................................................... 83
Change................................................................................................................................................. 83
Action required.................................................................................................................................... 83
Changed default settings for VA Assist Pro............................................................................................ 83
Reason for change............................................................................................................................... 83
Change................................................................................................................................................. 83
Action required.................................................................................................................................... 84
Abt Hover Tooltip text now left-aligned by default................................................................................. 84
Reason for change............................................................................................................................... 84
Change................................................................................................................................................. 84
Action required.................................................................................................................................... 84
Class definition template changed in standard browsers........................................................................ 84
Reason for change............................................................................................................................... 85
Change................................................................................................................................................. 85
Action required.................................................................................................................................... 85
Migrating from Version 8.5.1......................................................................................................................... 86
Parameters in the [log4s] stanza of the .INI file should not be enclosed in parentheses......................... 86
Reason for change............................................................................................................................... 86
Change................................................................................................................................................. 86
Action required.................................................................................................................................... 86
Time Zone Support Introduced................................................................................................................. 86
  Reason for change..................................................................................................................................................86
  Change........................................................................................................................................................................86
  Action required....................................................................................................................................................86
Prerequisite ICs changed in many IC packaging instructions................................................................. 87
  Reason for change..................................................................................................................................................87
  Change........................................................................................................................................................................87
  Action required....................................................................................................................................................87
String>>asInteger has been removed from EsLoggingFrameWorkApp................................................... 87
  Reason for change..................................................................................................................................................87
  Change........................................................................................................................................................................87
  Action required....................................................................................................................................................87
Migrating from Version 8.5.2......................................................................................................................... 88
  EsLogManager class>>error: and EsLogManager class>>info: have been removed from
  EsLoggingFrameWorkApp........................................................................................................................................ 88
  Reason for change..................................................................................................................................................88
  Change........................................................................................................................................................................88
  Action required....................................................................................................................................................88
Signal>>#description changed for compatibility with class-based Exceptions........................................ 88
  Reason for change..................................................................................................................................................88
  Change........................................................................................................................................................................89
  Examples.................................................................................................................................................................89
  Action required....................................................................................................................................................89
EsTimeZone class has changed......................................................................................................................... 90
  Reason for change..................................................................................................................................................90
  Change........................................................................................................................................................................90
  Action required....................................................................................................................................................90
Migrating from Version 8.6............................................................................................................................... 91
Deprecate and replace classes AbtBase64Coder and SstBase64Coder.................................................... 91

Reason for change........................................................................................................................................ 91

Change......................................................................................................................................................... 91

Action required........................................................................................................................................... 92

Encoding..................................................................................................................................................... 92

Decoding..................................................................................................................................................... 92

Other.......................................................................................................................................................... 92

Deprecate and replace incorrect Windows code page conversion API wrapper methods......................... 92

Reason for change........................................................................................................................................ 92

Change......................................................................................................................................................... 92

Action required.......................................................................................................................................... 93

Object>>#triggerEvent: added to AbtCLDTAdditions for portability.......................................................... 93

Reason for change......................................................................................................................................... 93

Change......................................................................................................................................................... 93

Action required.......................................................................................................................................... 93

Print methods added to Integer and Float in CLDT for portability........................................................... 93

Reason for change......................................................................................................................................... 93

Change......................................................................................................................................................... 93

Action required.......................................................................................................................................... 93

Deprecation.................................................................................................................................................. 94

Migration Tools.......................................................................................................................................... 96

Manager Import Tool................................................................................................................................ 96

Export/Import Passive Image Properties.................................................................................................... 97
Preface

Version 8.6.1, 2014

This edition applies to Version 8.6.1 of the 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 product or this document, address them to:
Instantiations, Inc.
Suite 1325B
Officers Row
Vancouver, WA 98661

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, 2014. All Rights Reserved
© Copyright International Business Machines Corporation 1994, 2002. All rights reserved.
This document covers migration from IBM VisualAge Smalltalk v1.0 and later plus all versions of VA Smalltalk through v8.6.1

Permission is granted to reproduce, print, copy, incorporate or otherwise distribute this document for the purpose of migrating from one version of VisualAge Smalltalk or VA Smalltalk to another as long as the included trademarks are acknowledged.

November 2014

Copyright Instantiations, Inc. 2008, 2014
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

The following terms are registered trademarks or trademarks of Instantiations, Inc. in the United States, other countries or both:

Instantiations
VA Smalltalk

The following terms are registered trademarks or trademarks of the International Business Machines Corporation in the United States, other countries or both:

AIX IBM
AIX/L IMS
C/370 International Business Machines
C/MVS Lotus Notes
C++/MVS Lotus Domino
CICS MQ
CICS/ESA MQSeries
Database 2 OS/390
DB2 OS/390 Native
DB2 Client Application Enablers SAA
DB2 Universal Database SQL/DS
Distributed Database Connection TX Series
Services/2
DRDA VisualAge

The following terms are trademarks of other companies:

Sun Solaris and Oracle
OpenSolaris
Netscape Netscape Inc.
Adobe Acrobat Adobe Inc.
Reader

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Microsoft, Windows, Windows NT, Windows Server, Windows Vista, Active Directory, OLE, OCX, Visual Basic, and the Windows Logo are either registered trademarks or trademarks of Microsoft Corporation in the United States, other countries or both.

UNIX is a registered trademark of The Open Group in the United States, other countries or both.
Other company and product names mentioned herein may be the trademarks or service marks of their respective owners.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred.
About this book

This book covers upgrading your existing applications from IBM VisualAge Smalltalk Enterprise Version 3.0 through 6.0.4, and previous versions of VA Smalltalk, to the current VA Smalltalk version. You should review and understand the following material, as appropriate, prior to loading your existing applications into the current version of 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 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?

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

- Migration Tools
- 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

In addition, the chapters of the book have been reordered so you can read the book 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 VA Smalltalk. Information applicable to a specific Windows environments is noted accordingly.

Tips and environment-specific information are flagged with icons:

- Shortcut techniques and other tips
- VA Smalltalk for Windows
- VA Smalltalk for UNIX platforms

These highlighting conventions are used in the text:
### Highlight style

<table>
<thead>
<tr>
<th>Highlight style</th>
<th>Used for</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Boldface</strong></td>
<td>New terms the first time they are used</td>
<td>VA Smalltalk uses <strong>construction from parts</strong> to develop software by assembling and connecting reusable components called <strong>parts</strong>.</td>
</tr>
<tr>
<td></td>
<td>Items you can select, such as push buttons and menu choices</td>
<td>Select <strong>Add Part</strong> from the <strong>Options</strong> pull-down. Type the part's class and select <strong>OK</strong>.</td>
</tr>
<tr>
<td><strong>Italic</strong></td>
<td>Special emphasis</td>
<td>Do <strong>not</strong> save the image.</td>
</tr>
<tr>
<td>Titles of publications</td>
<td>Refer to the <em>VA Smalltalk User's Guide</em>.</td>
<td></td>
</tr>
<tr>
<td>Text that the product displays</td>
<td>The status area displays <em>Category: Data Entry</em>.</td>
<td></td>
</tr>
<tr>
<td>VA Smalltalk programming objects, such as <em>attributes, actions, events, composite parts, and script names</em></td>
<td>Connect the window's <em>aboutToOpenWidget</em> event to the <em>initializeWhereClause</em> script.</td>
<td></td>
</tr>
</tbody>
</table>
| **Monospace font** | VA Smalltalk scripts and other examples of Smalltalk code | `doSomething
| | `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 fax comments to (503)649-3836 (call first), email comments to vast-support@instantiations.com or post comments and technical questions to the VA Smalltalk community support forum. A link to the VA Smalltalk Forum can be found at “Support Forum” and instructions on what information to include with technical problems can be found at “Technical Support” on our VA Smalltalk web page. You can also access the VA Smalltalk Forum from any of the VA Smalltalk Help menus.

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 VA Smalltalk, such as Version 7.5 or Version 8.0, into the default directory location, any older installations of 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 VA Smalltalk, 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 VA Smalltalk.
- 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

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 a previous version of VA Smalltalk, then after the successful installation of your new version of 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 VA Smalltalk as the new development library.

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

- merge your applications from the existing development library into the new development library
- merge all applications from the new development library into the existing development library

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 to the current VA Smalltalk 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 VA Smalltalk Base User's 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 VA Smalltalk library to your existing VisualAge Smalltalk or VA Smalltalk 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 VA Smalltalk Library Manager, the installation process creates a new library (for example, mgr86.dat). Merging code from the current version of VA Smalltalk into your existing library involves running the importer utility to import the newer product code into your library. You can then resume work using your current 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:

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

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 VA Smalltalk User's 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 VA Smalltalk Base User's 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.
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 VisualAge: 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 VisualAge: Database, Static DB2 Support feature, the Dynamic to Static menu choice is added.

Refer to the VA Smalltalk User's 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 VA Smalltalk 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:
Migration Guide

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.

   ![Database Migration Window](image)

   **Migration Specifications**
   
   **Application To Migrate**
   
   **Target Connection alias**
   
   **Parts**
   
   ![List of Parts](image)

   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 *VA Smalltalk 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 *VA Smalltalk Base User's 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:errorMessage:`

**Methods in ObjectLoader**

- `loadFromFileHandle:atOffset:`
- `loadFromFileHandle:atOffset:errorMessage:`

**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.

``` Smalltalk
| 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.

``` Smalltalk
| aCfsFileDescriptor someObject |

aCfsFileDescriptor := CfsFileDescriptor open: 'someobj.swp' oflag: ORDONLY.
aCfsFileDescriptor isCfsError ifTrue: [^self error: 'Cannot open file'].
someObject := ObjectLoader new
loadFrom: someObject
fromFileHandle: aCfsFileDescriptor
atOffset: 0.
aCfsFileDescriptor close.
System message: someObject printString.
```
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, `AbtAppBlldrPart` had the following pool dictionaries defined: `CwConstants`, `AbtCwConstants`, and `EwConstants`. If one of your classes subclasses from `AbtAppBlldrPart` (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 `AbtAppBlldrVisual`, `AbtAppBlldrView`, and probably in other classes. If your classes subclass directly from `AbtAppBlldrNonVisual` or `AbtAppBlldrPart`, 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:
   ```smalltalk
   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
... AbtAppBlldrPart
... 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 `AbtAppBlldrVisual` but not `AbtAppBlldrNonVisual`. 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 \textit{OSObject} became an abstract class which should not be directly instantiated or subclassed. An \textit{OSObject} subclass called \textit{OSPtr} now contains the function previously available in \textit{OSObject}. Additionally, \textit{OSPtr} contains a subclass called \textit{AbtForeignOSObject} which supports code page conversion. Existing applications that currently use \textit{OSObject} subclasses or instances to share data with external functions need to be migrated to use \textit{OSPtr} or another suitable \textit{OSObject} subclass.

**Migrating \textit{OSObject} subclasses**

To migrate \textit{OSObject} subclasses, first determine which classes need to be migrated:

1. Load existing applications that use \textit{OSObject} instances or subclasses to interface with external functions.
2. In the System Transcript, select \textbf{Browse Hierarchy} from the \textbf{Tools} menu. A prompter opens, requesting a class name.
3. Type \texttt{OSObject} and select \textbf{OK}. A hierarchy browser opens, showing the class \textit{OSObject} and its direct subclasses.
   - The classes \textit{OSImmediate}, \textit{OSPtr}, and \textit{OSStorage} are base Smalltalk classes which you should \textit{not} migrate.
   - The remaining \textit{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
   
   \begin{verbatim}
   OSObject subclass: #MyVersion3Class
   instanceVariableNames: ""
   classVariableNames: ""
   poolDictionaries: ""
   \end{verbatim}

   to
   
   \begin{verbatim}
   OSPtr subclass: #MyVersion3Class
   instanceVariableNames: ""
   classVariableNames: ""
   poolDictionaries: ""
   \end{verbatim}

   If the class being migrated requires code page conversion support, use \textit{AbtForeignOSObject} as a superclass as follows:
   
   \begin{verbatim}
   AbtForeignOSObject subclass: #MyVersion3Class
   instanceVariableNames: ""
   classVariableNames: ""
   poolDictionaries: ""
   \end{verbatim}

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

**Migrating code that instantiates \textit{OSObject} directly**

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

1. From the System Transcript, select \textbf{Browse References} from the \textbf{Tools} menu.
2. In the displayed prompter, type \texttt{OSObject} and select \textbf{OK}. A References browser opens.
3. Scroll through the list of methods that reference \textit{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.

<table>
<thead>
<tr>
<th>Connection</th>
<th>Source readOnly or target readOnly</th>
<th>Target readOnly or source nil</th>
<th>V3.0 result</th>
<th>VA Smalltalk result</th>
<th>Migration restores V3.0 result</th>
<th>Needs your attention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent A &gt; Wrapper A</td>
<td>true</td>
<td>true</td>
<td>No alignment</td>
<td>No alignment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A unidirectional connection from attribute A of the parent part to attribute A of the view wrapper</td>
<td>true</td>
<td>false</td>
<td>Parent A &gt; Wrapper A</td>
<td>Parent A &gt; Wrapper A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>No alignment</td>
<td>No alignment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>Parent A &gt; Wrapper A</td>
<td>Parent A &gt; Wrapper A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wrapper A &gt; Parent A</td>
<td>true</td>
<td>true</td>
<td>No alignment</td>
<td>No alignment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A unidirectional connection from attribute A of the view wrapper to attribute A of the parent part</td>
<td>true</td>
<td>false</td>
<td>No alignment</td>
<td>Wrapping A &gt; Parent A The value of A in the parent part is set to the value of A in the view wrapper. X X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>Parent A &gt; Wrapper A</td>
<td>No alignment No alignment takes place. X X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>Parent A &gt; Wrapper A</td>
<td>Wrapping A &gt; Parent A The value of A in the parent part is set to the value of A in the view wrapper. X X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parent A &lt;&gt; Wrapper A</td>
<td>true</td>
<td>true</td>
<td>No alignment</td>
<td>No alignment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A bidirectional connection from attribute A of the parent part to attribute A of the view wrapper</td>
<td>true</td>
<td>false</td>
<td>Parent A &gt; Wrapper A</td>
<td>Parent A &gt; Wrapper A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>No alignment</td>
<td>Wrapping A &gt; Parent A The value of A in the parent part is set to the</td>
<td>X X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 `AbtAppBlldrView`, `selectedItemFromTree` is the domain object, and `myContainerIconTree` is an instance of 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:

```
def 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.
```
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 begin 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 *IBM Smalltalk User's 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`
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 \textit{DsGlobalReference}:

- \textit{shadow}
- \textit{repString}
- \textit{repStringOn: \text{aStream}}
- \textit{objectRefString: \text{aString}}
- \textit{objectRefStringOn: \text{aStream}}

\textbf{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 \textit{String}:

- \textit{sstGlobalIn: \text{aSpace}}
- \textit{sstGlobalIn: \text{aSpace context: \text{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:

- \textit{export: \text{object}}
- \textit{export: \text{object as: \text{objectId}}}
- \textit{unexportId: \text{key}}
- \textit{unexportObject: \text{object}}

Example code to make the server object space's date externally known, with the ID \textit{#serverDate} includes--
\begin{verbatim}
serverObjectSpace export: Date today as: #serverDate
\end{verbatim}

The class \textit{SstRemoteReference} provides the following class method API to access remote objects by symbolic references:

- \textit{export: \text{object in: \text{space}}}
- \textit{for: \text{object in: \text{space}}}

The following example client code returns a remote reference to the object identified by \textit{#serverDate} in \textit{serverObjectSpace}:

\begin{verbatim}
SstRemoteReference for: #serverDate in: serverObjectSpace
\end{verbatim}

\textbf{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 \textit{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`:

```plaintext```
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 provide 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*
Migration Guide

- 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**
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

```smalltalk
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 circumstances 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

```smalltalk
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

```smalltalk
firstName: aString
```

This is the incorrect syntax. The event `firstName` is being signalled with the current value, not the newly assigned value.
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
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 snapshots, 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 begin 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:

- 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

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 VA Smalltalk Base User's 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 VA Smalltalk User's 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.

<table>
<thead>
<tr>
<th>Control file information</th>
<th>Support in Version 6.0 and VA Smalltalk</th>
</tr>
</thead>
<tbody>
<tr>
<td>header information</td>
<td>Supported with changes</td>
</tr>
<tr>
<td>include</td>
<td>Supported</td>
</tr>
<tr>
<td>includeFeatureMap</td>
<td>New in Version 5.5</td>
</tr>
<tr>
<td>import</td>
<td>Not supported</td>
</tr>
<tr>
<td>mapname</td>
<td>Supported with changes</td>
</tr>
</tbody>
</table>

**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:
```
```

**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 importer utility.

**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:
- `mynmap' 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"
```
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. VisualAge Smalltalk V6.0 and VA Smalltalk installs an importer program, `importer.exe`, in the `<vast>\inst` directory of machines with the client product installed and the manager product installed.

The importer accepts the following parameters:

- `-z.target=`
  accepts the location of the target library. This is the customer's V6.0 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. All *.dat files in the directory will be processed.
  
  **Note:**
  Either `-z.source` or `-z.sourcedir` must be specified, but never both.

- `-z.silent`
  Runs with no user interface.

After the utility runs, check for non-zero exit codes. The importer.exe will end with a 0 (zero) exit code if the import was successful. If the import was not successful, the utility will end with a non-zero exit code.

The following are examples of how to run the utility:

```
importer.exe -z.target=zot::d:\vast55\manager\mgr55.dat -z.source=e:\tmp\myfeat.dat
importer.exe -z.target=zot::d:\Program_Files\vast55\manager\mgr55.dat -z.source=e:\tmp\myfeat.dat
importer.exe -z.target=zot::d:\vast55\manager\mgr55.dat -z.sourcedir=e:\tmp\featureDir
```

**Note:**
With File I/O only one person may be using the library manager 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.

### 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 SstHttpServletRequest instead. Existing references to SstHttpCgiServlet should be changed to make use of SstHttpServletRequest 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 SstHttpServletRequestEngine.

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 VA Smalltalk Base Programer's Reference. The priority is not forced to a priority of 3 and the wait message send is not converted to a Processor>>#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
Migration Guide

- 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 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
Migration Guide

- 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, 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:

```smalltalk
#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*. 
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:

```smalltalk
aboutToBeSwitchedTo: aComp
aComp setFocus
```

could be rewritten as:

```smalltalk
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):

```ini
[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"

<table>
<thead>
<tr>
<th></th>
<th>V6.0 and earlier</th>
<th>6.0.1</th>
<th>V6.0.2 and later</th>
</tr>
</thead>
<tbody>
<tr>
<td>badArrayIdentity</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>stringLiteralIdentity</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>mixedArray</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>poolDictionaryModification</td>
<td>'x'</td>
<td>'x'</td>
<td>'y'</td>
</tr>
</tbody>
</table>
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 and VisualAgeVendorExtensionsApp
- Server Smalltalk (SST) feature definition changes
- Merge ENVI/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’s 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 ENVI/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):
Generally, the idea was that the \textit{ST: Client, SST} feature included those maps needed to build a Server Smalltalk Client application and the \textit{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 \textit{ST: Client, SST} feature has been replaced with a suite of features and the \textit{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 \textit{ST: Client, SST} feature are:

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

\textbf{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:milliseconds:` has been changed to better align with the internal representation of AbtTimestamp. Previously the value of the `milliseconds:` argument could range from 0 – 9999999 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:
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:

- 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 begin 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 SmalltalkKernel 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:
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:

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

**`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 3rd-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
  anInteger atRandom

  GRPlatform current newRandom randomFrom: aCollection
  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
  aCollection endsWith: aSubCollection

  aCollection beginsWithSubCollection: aSubCollection
  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:

```smalltalk
aDuration milliseconds
```
to:

```smalltalk
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 \texttt{EwxIconManager} (which was the only correct implementation):

\texttt{EwIconManager}
\texttt{EwButtonIconManager}
\texttt{WkIconManager}
\texttt{EwTriangleIconManager}
\texttt{WbIconManager}
\texttt{EwxIconManager}

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

\texttt{EwButtonIconManager} and \texttt{EwTriangleIconManager} provide specific support for tree hierarchy indicators.\texttt{WkIconManager} (from \texttt{WidgetKit/Controls - Runtime}) and \texttt{WbIconManager} (from \texttt{WindowBuilderPro – Runtime}) have been deprecated and are empty classes kept only to satisfy any references to the classes in customer code.

\texttt{EwxIconManager} provides icon management specific to the \texttt{EwExamples} application.

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

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

**Action required**

If you use, extend, or subclass any of the \texttt{...IconManager} classes, you may need to make changes to your application:

- If you have subclasses of any class named \texttt{...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 \texttt{...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 \texttt{...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 \texttt{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.
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 begin 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 recomputed 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.

<table>
<thead>
<tr>
<th>Old Stanza</th>
<th>Old Keyword</th>
<th>New Stanza</th>
<th>New Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>fplevel</td>
<td>AbtCommonProductInstallerApp</td>
<td><em>same</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AbtCommonProductInstallerApp</td>
<td></td>
</tr>
<tr>
<td>EvaluationLogName</td>
<td></td>
<td>EtBaseTools</td>
<td><em>same</em></td>
</tr>
<tr>
<td>AllowOutOfScopeReferences</td>
<td></td>
<td><em>deleted – unused</em></td>
<td></td>
</tr>
<tr>
<td>NLS Config</td>
<td>ignoreCatalogAliasing</td>
<td>AbtNlsKernelApp</td>
<td><em>same</em></td>
</tr>
<tr>
<td>VAST 5.0</td>
<td>installationPath</td>
<td>AbtCommonProductInstallerApp</td>
<td><em>same</em></td>
</tr>
<tr>
<td>Compatibility</td>
<td></td>
<td>AbtCommonProductInstallerApp</td>
<td></td>
</tr>
</tbody>
</table>

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’s 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 empty ('') or had trailing directory separator characters that were not consistent with what was expected.

**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:

```smalltalk
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:

```smalltalk
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 begin 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`. 
Migration Guide

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's 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 use of multi-line text within hover tooltips are 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 |

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:

```plaintext
NameOfSuperclass subclass: #NameOfClass
    instanceVariableNames: 'instVarName1 instVarName2'
    classVariableNames: 'ClassVarName1 ClassVarName2'
    poolDictionaries:"
```

but in an image with VA Assist loaded it was:

```plaintext
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


**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 begin 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.

``` Smalltalk
[(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: Error 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).
# Deprecation

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.

<table>
<thead>
<tr>
<th>Original Method</th>
<th>Replacement Method</th>
<th>Deprecated in Version</th>
<th>Removed in Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbtBase64Coder class&gt;&gt;current</td>
<td>Base64CoderMime class&gt;&gt;current</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td>AbtBase64Coder class&gt;&gt;reset</td>
<td>Base64CoderMime class&gt;&gt;reset</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td>Color&gt;&gt;hue:saturation:brightness:</td>
<td>Color&gt;&gt;h:s:v:</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td>Color&gt;&gt;abtInverse</td>
<td>Color&gt;&gt;inverse</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td>Color&gt;&gt;caAsInverse</td>
<td>Color&gt;&gt;inverse</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td>CompiledMethod&gt;&gt;stsDeveloper</td>
<td>CompiledMethod&gt;&gt;developer</td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td>CwScintillaEditor&gt;&gt;getStyleBits</td>
<td>CwScintillaEditor&gt;&gt;getStyleBitsNeeded</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td>CwScintillaEditor&gt;&gt;getStyleBitsNeeded</td>
<td>n/a</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td>CwScintillaEditor&gt;&gt;setStyleBits:</td>
<td>n/a</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td>DateAndTime class&gt;&gt;year:</td>
<td>DateAndTime class&gt;&gt;year:</td>
<td>8.5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Migration Version</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td><code>EsLogManager class&gt;&gt;info:locationInfo:</code></td>
<td><code>EsLogManager class&gt;&gt;logInfo:locationInfo:</code></td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>EsLogManager class&gt;&gt;warn:locationInfo:</code></td>
<td><code>EsLogManager class&gt;&gt;logWarn:locationInfo:</code></td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>EsLogManager class&gt;&gt;debug:locationInfo:</code></td>
<td><code>EsLogManager class&gt;&gt;logDebug:locationInfo:</code></td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>EsLogManager class&gt;&gt;error:locationInfo:</code></td>
<td><code>EsLogManager class&gt;&gt;logError:locationInfo:</code></td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>ExceptionalEvent&gt;&gt;ancestorOf:</code></td>
<td><code>ExceptionalEvent&gt;&gt;handles:</code></td>
<td>8.0.2</td>
<td></td>
</tr>
<tr>
<td><code>ExceptionalEventCollection&gt;&gt;ancestorOf:</code></td>
<td><code>ExceptionalEventCollection&gt;&gt;handles:</code></td>
<td>8.0.2</td>
<td></td>
</tr>
<tr>
<td><code>Object&gt;&gt;abtIsProcess</code></td>
<td><code>Object&gt;&gt;isProcess</code></td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>Object&gt;&gt;instVarAtName:</code></td>
<td><code>Object&gt;&gt;instVarNamed:</code></td>
<td>8.0.2</td>
<td></td>
</tr>
<tr>
<td><code>Object&gt;&gt;instVarAtName:put:</code></td>
<td><code>Object&gt;&gt;instVarNamed:put:</code></td>
<td>8.0.2</td>
<td></td>
</tr>
<tr>
<td><code>OSCall&gt;&gt;initCommonControls</code></td>
<td><code>OSCall&gt;&gt;initCommonControlsEx</code></td>
<td>8.5</td>
<td></td>
</tr>
<tr>
<td><code>OSCall&gt;&gt;multiByteToWideChar:dwFlags:lpMultiByteStr:cbMultiByte:lpWideCharStr:cchWideChar:</code></td>
<td><code>OSCall&gt;&gt;multiByteToWideChar:dwFlags:lpMultiByteStr:cbMultiByte:lpWideCharStr:cchWideChar:</code></td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>OSCall&gt;&gt;setMessageQueue:</code></td>
<td>n/a</td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>OSCall&gt;&gt; wideCharToMultiByte:dwFlags:lpWideCharStr:cchWideChar:lpMultiByteStr:cbMultiByte:lpDefaultChar:lpUsedDefaultChar:</code></td>
<td><code>OSCall&gt;&gt; wideCharToMultiByte:dwFlags:lpWideCharStr:cchWideChar:lpMultiByteStr:cbMultiByte:lpDefaultChar:lpUsedDefaultChar:</code></td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>Pragma&gt;&gt;analogousCodeTo:</code></td>
<td>n/a</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>Pragma&gt;hasLiteral:</code></td>
<td>n/a</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>Pragma&gt;&gt;hasLiteralSuchThat:</code></td>
<td>n/a</td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>ScaledDecimal&gt;&gt;asDecimal</code></td>
<td><code>ScaledDecimal&gt;&gt;asScaledDecimal</code></td>
<td>8.5</td>
<td></td>
</tr>
<tr>
<td><code>ScaledDecimal&gt;&gt;precision:</code></td>
<td><code>ScaledDecimal&gt;&gt;privatePrecision:</code></td>
<td>8.5</td>
<td></td>
</tr>
<tr>
<td><code>ScaledDecimal&gt;&gt;scale:</code></td>
<td><code>ScaledDecimal&gt;&gt;privateScale:</code></td>
<td>8.5</td>
<td></td>
</tr>
<tr>
<td><code>StsBase64Coder class&gt;&gt;new</code></td>
<td><code>Base64CoderMime class&gt;&gt;new</code></td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>StsColorText class&gt;&gt;exiting</code></td>
<td>n/a</td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>StsColorText class&gt;&gt;fontList</code></td>
<td><code>StsCodeFontManager&gt;&gt;fontList</code></td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>StsColorText class&gt;&gt;fontList:</code></td>
<td><code>StsCodeFontManager&gt;&gt;fontList:</code></td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>StsColorText class&gt;&gt;fontName</code></td>
<td><code>StsCodeFontManager&gt;&gt;fontName</code></td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>StsColorText class&gt;&gt;preStartUp</code></td>
<td>n/a</td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td><code>StsPowerTools class&gt;&gt;currentColorScheme</code></td>
<td><code>StsPowerTools class&gt;&gt;currentEditorTheme</code></td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>StsPowerTools class&gt;&gt;currentColorScheme</code></td>
<td><code>StsPowerTools class&gt;&gt;currentEditorTheme</code></td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>StsPowerTools class&gt;&gt;currentColorSchemeMap</code></td>
<td><code>StsPowerTools class&gt;&gt;currentEditorThemeMap</code></td>
<td>8.6.1</td>
<td></td>
</tr>
<tr>
<td><code>WbApplication class&gt;&gt;isWin32</code></td>
<td>n/a</td>
<td>8.6.1</td>
<td></td>
</tr>
</tbody>
</table>
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**

Manager Import 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. The *importer* program is the tool that provides this function. Both the client and the manager product installation processes install the importer program, importer.exe, in the `<vast>\inst` directory.

The *importer* is most useful when installing a new version of VA Smalltalk 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 newest version of each configuration map.

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. All *.dat files in the directory will be processed.

- **-z.silent** runs with no user interface.

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.

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

The following are some examples of how to run the utility:

```
importer.exe -z.target=localhost::d:\vamgr\manager\mgr80.dat -z.source=e:\tmp\myfeat.dat
importer.exe -z.target=c:\Program Files\Instantiations\VA Smalltalk\8.5\manager\mgr80-working.dat" -z.source=c:\Program Files\Instantiations\VA Smalltalk\8.5\manager\mgr80.dat -z.sourcedir=e:\tmp\featureDir -z.silent
```
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
   ```smalltalk
   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:
   ```smalltalk
   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.