JIS Interface Server:

Release Notes

Version 9.3

September 2026

This document applies to JIS Interface Server Version 9.3 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright (c) 1992-2026 Software GmbH, Darmstadt, Germany and/or their suppliers. All rights reserved.

The name Software GmbH and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH. Other company and product names mentioned herein may be trademarks of their respective owners.

Document ID: JIS-GS-930-20260923

Table of Contents

Overview **1**
About JIS Interface Server 1
Organization 1

JIS Interface Server 9.3 Release Notes **3**
About this release 3
Release highlights 3
Installation and upgrade information 4
Migrating BDE application databases to SQLite 4
Licensing 4
Installation and Upgrade Instructions 5
Pack JIS Applications from Previous Version of JIS Interface Server 5
Install JIS Interface Server 9.3 6
Configure JIS Interface Server 9.3 to Run on an External Java Installation 6
Unpack the Jpack in JIS Interface Server 9.3 6
Recompile the JIS Interface Server Application 7
Recompile any Java Extensions 7
Create a Runtime Installation 7
Java client 7
XHTML client 8
Install the JIS Runtime Installation 8
Test the Runtime 9
Configuration and encoding 9
General Data Protection Regulation 10
New features and enhancements 10
Jakarta and embedded web-runtime modernization 10
Perl toolchain modernization 10
SQLite application database and legacy migration 11
Software AG license harmonization 11
Enhanced Content Security Policy support 11
Protected redirect URLs 11
Security and hardening 12
Detailed description of fixes 12
Embedded server and HTTPS 12
Java, JRO, and Java Web Start 13
Application database and packaging 13
Licensing 13
Screen generation and browser behavior 14
Build, signing, and diagnostics 14
Included component updates 15
New and changed settings 16
Compatibility and behavior changes 17

JIS Interface Server 9.2.5 Release Notes 18
Installation & Upgrade Information 18
Supported Platforms 18
Recommended Configurations 18
ACE 18
Clients 19
Java Web Start Clients 19
XHTML Clients 19
Application Server Deployment 20
Standalone Server 20
Installation and Upgrade Instructions 21
Pack JIS Applications from Previous Version of JIS Interface Server 22
Install JIS Interface Server 9.2.5 22
Configure JIS Interface Server 9.2.5 to Run on an External Java Installation 23
Unpack the Jpack in JIS Interface Server 9.2.5 23
Recompile the JIS Interface Server Application 23
Recompile any Java Extensions 24
Create a Runtime Installation 24
Install the JIS Runtime Installation 25
Test the Runtime 25
General Data Protection Regulation 26
Retirement of Product Components 26
New Features Included in JIS 9.2.5 26
Updated Operating System Support 26
Java Version 17 Support 26
Azul Java 17 Support 27
Tomcat 10 Support 27
Server Configuration Properties 27
HTTP Server Configuration Properties 28
MakeJar Server Configuration Properties 29
Application Configuration Properties 32
Detailed Description of Fixes Included in JIS 9.2.5 33
Limitations of JIS 9.2.5 33
JIS-2413 / SI-466468 - ContentSecurityPolicy vulnerability 34

JIS Interface Server 9.2.4 Release Notes 35
Installation & Upgrade Information 35
Supported Platforms 35
Recommended Configurations 35
ACE 35
Clients 36
Java Web Start Clients 36
XHTML Clients 36
Application Server Deployment 37
Standalone Server 37
Installation and Upgrade Instructions 38
Pack JIS Applications from Previous Version of JIS Interface Server 39
Install JIS Interface Server 9.2.4 39

Configure JIS Interface Server 9.2.4 to Run on an External Java Installation 40
Unpack the Jpack in JIS Interface Server 9.2.4 40
Recompile the JIS Interface Server Application 41
Recompile any Java Extensions 41
Create a Runtime Installation 42
Install the JIS Runtime Installation 43
Test the Runtime 43
General Data Protection Regulation 43
Retirement of Product Components 43
Java Client Applet Support 44
Legacy Unix Support 44
New Features Included in JIS 9.2.4 44
HTTP RelativeRedirectAllowed Enabled by Default 44
Upgrade to Jetty 9.4.53 45
Server/Host SSL Protocol & Cipher Suite Settings 45
Content-Type Header Charset Setting 45
Run JAM as a JNLP Application 45
Jetty Logging Setting 45
128-Bit Session IDs 45
OS Level TCPKeepAlive Setting 46
XHTML RedirectionProxy with SupportHTTPSOnly 46
Jarsigner Command Line Option Settings 46
Unbundled JVM or JDK 46
IzPack Upgrade to Version 5.2.3 46
New Server Configuration Properties 46
New HTTP Server Configuration Properties 47
New MakeJar Server Configuration Properties 48
New Application Configuration Properties 52
Detailed Description of Fixes Included in JIS 9.2.4 53
ACE 57
JIS-2157 / PI-1356981 - JIS Administrator not showing ini settings JIS-2162 - The jrodefaults.ini file is not installed in the runtime installation JIS-2397 / SI-460216 - Jacada Administrator - Connection Closed error. JIS-2399 - jrodefaults.ini is not included in the JIS runtime installation 57
JIS-2382 / SI-453775 - Need impact on the product logging due to Log4J vulnerability|Create a patch for 9.2.2 and 9.2.3 that will exclude the JMSAppender class. JIS-2383 - Remove JMSAppender class from Natural Parser log4j.jar 57
JIS-2451 / SI-485344 - ACE.exe digital cert expired JIS-2452 - Update JIS code signing certificate for 2022-2023 57
JIS-2588 / SI-567394 - Jacada Cert Issue JIS-2590 - Add properties for additional jarsigner command line options 57
JIS-2602 / SI-569157 - JIS 9.2.4 beta release attachment JIS-2608 - Support for Java command line utils on Java 17 JIS-2609 - Upgrade IzPack for Java 17 support JIS-2610 - Update batch files for unbundled JVM or JDK 58
JAM 58
JIS-2357 / PI-5457849 - Blackduck Code scan JIS-2362 - Validate DebugLevel in AbstractJAMServlet 58

JIS-2397 / SI-460216 - Jacada Administrator - Connection Closed error. JIS-2398 - Fix ClassCastException thrown from JAM while fetching INI file params. JIS-2400 - Improve error handling for missing jrodefaults.ini file 58
JIS-2408 / SI-463941 - Source scan Issues in jam.jar and jacada Files JIS-2446 - Investigate and if possible remove instances of deserialization of untrusted data . . . 59
JIS-2427 / SI-474156 - Error with admin.jnlp JIS-2433 - Add support for running JAM as a JNLP application 59
JIS-2451 / SI-485344 - ACE.exe digital cert expired JIS-2452 - Update JIS code signing certificate for 2022-2023 59
JIS-2456 / SI-489457 - Java update and JIS JIS-2458 - Update JIS Java build version 59
Oracle Java 8u351 introduced a change in the way that signed jar files are treated for Java Web Start / JNLP applications. Starting with Java 8u351, Jar files signed with SHA-1 algorithms are now treated as unsigned, preventing them from being loaded via JNLP. This, in turn prevents them from being used in JIS Java Client applications started via Java Web Start / JNLP. The JIS clbase and jam jars are now signed with SHA-256 algorithms, which are compatible with Java 8u351 59
JIS-2580 / SI-564462 - Need help getting admin tool working on the Linux servers. JIS-2595 - Modify the method for Deserialization 60
Java client 60
JIS-2352 / PI-1458958 - Application Menu JIS-2522 - Handle NullPointerException when running a JWS Java Client Application with RunInsideBrowser=TRUE 60
JIS-2408 / SI-463941 - Source scan Issues in jam.jar and jacada Files JIS-2446 - Investigate and if possible remove instances of deserialization of untrusted data . . . 60
JIS-2451 / SI-485344 - ACE.exe digital cert expired JIS-2452 - Update JIS code signing certificate for 2022-2023 60
JIS-2456 / SI-489457 - Java update and JIS JIS-2458 - Update JIS Java build version 60
Oracle Java 8u351 introduced a change in the way that signed jar files are treated for Java Web Start / JNLP applications. Starting with Java 8u351, Jar files signed with SHA-1 algorithms are now treated as unsigned, preventing them from being loaded via JNLP. This, in turn prevents them from being used in JIS Java Client applications started via Java Web Start / JNLP. The JIS clbase and jam jars are now signed with SHA-256 algorithms, which are compatible with Java 8u351 60
JIS-2479 / SI-503157 - Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) (CWE ID 80) JIS-2519 - Sanitize URL used for generating JAM JNLP file 61
JIS-2548 / SI-551097 - Penetration Scan has Identified Predictable Session Tokens Vulnerability JIS-2569 - Upgrade Session IDs to 128 bits using UUID 61
JIS-2602 / SI-569157 - JIS 9.2.4 beta release attachment JIS-2608 - Support for Java command line utils on Java 17 61
Server 61
JIS-2029 / PI-1292251 - 9.2.2 - Compile Error after upgrading JIS-2224 - JNLP Servlet log should be created in the directory specified by RtLogDir or -l startup option 61
JIS-2220 / PI-5391880 - 404 error when migrating to new upgraded prod JIS-2221 - Add support for server / host SSL protocol and cipher suite settings 62

JIS-2318 / PI-5441448 - Paste with a emdash causes disconnect JIS-2340 - Remove em-dash and en-dash unicode characters from data sent to host. 48
JIS-2338 / PI-5444847 - Multi-browser Support XhtmlSubapplGennerator:error Null Pointer JIS-2343 - Add checks for null pointers and improve logging 62
JIS-2357 / PI-5457849 - Blackduck Code scan JIS-2361 - CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page JIS-2363 - Remove dependence on URL query string for JIS session information JIS-2366 - Update embedded Jetty to version 9.4.44 62
JIS-2368 / SI-450620 - Null pointer exception when using JNLP JIS-2371 - Revert access change for RunTimeApplication InitBeforeFirstSubAppl methods 62
JIS-2375 / SI-452878 - Hotfix 3216 and Session closed message JIS-2380 - KeepAlive requests are failing on chrome when language is unset 63
JIS-2377 / SI-452781 - Content-Type charset setting JIS-2424 - Add JIS server property setting to allow configuring Content-Type header charset 63
JIS-2384 / SI-454668 - Web-Cache Poisoning JIS-2474 - Add RelativeRedirectAllowed to Jetty config 63
JIS-2395 / SI-461620 - Client is Reporting Latency Since Deploying Multi-Browser Changes JIS-2402 - Remove extraneous debug logging from the com.jacada.jis.runtime.server.http.ActionHandler class 63
JIS-2396 / SI-462140 - Session timeout message after 60 seconds sitting idle JIS-2401 - Fix parsing of URL query string for JBS parameter 63
JIS-2408 / SI-463941 - Source scan Issues in jam.jar and jacada Files JIS-2489 / SI-503160 - Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG) (CWE ID 338) JIS-2410 - Allow JRE to specify SecureRandom algorithm JIS-2446 - Investigate and if possible remove instances of deserialization of untrusted data 63
JIS-2417 / SI-468446 - Jacada 9.2.3 - Jacada Claim JIS-2420 - Improve debug logging JIS-2421 - Global variables for screen interpreter only initialized for first session. . 50 JIS-2427 / SI-474156 - Error with admin.jnlp JIS-2433 - Add support for running JAM as a JNLP application 64
JIS-2436 / SI-476930 - Jacada Production Issue JIS-2439 - Fix JMX Command Line Operations suspend method 64
JIS-2437 / SI-477423 - Increase trace level in JBS file JIS-2441 - Enable setting of SessionCoreDump properties via server INI in addition to appl ini 64
JIS-2451 / SI-485344 - ACE.exe digital cert expired JIS-2452 - Update JIS code signing certificate for 2022-2023 65
JIS-2460 / SI-491960 - Jetty Log is growing very large on server JIS-2461 - Add support for configuring jetty logging 65
JIS-2471 / SI-498702 - CTE specs for BBI JIS-2472 - Add jetty version to JIS server log 65
JIS-2475 / SI-501334 - Delta Air Lines - JIS Vulnerability Scan JIS-2476 / SI-503192 - J2EE Bad Practices: Use of System.exit() (CWE ID 382) JIS-2515 - Mitigate use of System.exit() (CWE ID 382) 65
JIS-2477 / SI-503193 - Improper Resource Shutdown or Release (CWE ID 404) JIS-2497 - Fixes for improper resource shutdown (CWE ID-404) 65
JIS-2479 / SI-503157 - Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) (CWE ID 80) JIS-2519 - Sanitize URL used for generating JAM JNLP file 66

JIS-2485 / SI-503161 - Use of a Broken or Risky Cryptographic Algorithm (CWE ID 327) JIS-2531 / SI-520395 - Production Issue - Users receiving blank screen JIS-2502 - Add logging for debugging redirection proxy JIS-2516 - Replace DES with AES algorithm 66
JIS-2486 / SI-503162 - External Control of File Name or Path (CWE ID 73) JIS-2512 - Fix filename for vulnerability 66
JIS-2487 / SI-503163 - Deserialization of Untrusted Data (CWE ID 502) JIS-2544 - Resolve deserialization issue (CWE Id-502) 66
JIS-2491 / SI-503168 - Improper Restriction of XML External Entity Reference (CWE ID 611) JIS-2514 - Disable external entity reference in the XML document builder 66
JIS-2494 / SI-503172 - Selection of Less-Secure Algorithm During Negotiation ('Algorithm Downgrade') (CWE ID 757) JIS-2511 - Disable TLS1.1 67
JIS-2495 / SI-503507 - Library XHTML Template Extensions Aren't Merged JIS-2496 - Fix merging of library user extensions 67
JIS-2504 / SI-505992 - Security Vulnerabilities for TLS Protocol and Cipher Suites JIS-2520 - Apply SupportedProtocols and SupportedCipherSuites properties to second JIS SSL server port 67
JIS-2527 / SI-518330 - Issues that reported in Blackduck scan to be fixed JIS-2528 - Upgrade embedded Jetty to version 9.4.52 67
JIS-2534 / SI-523094 - JIS Secure Telnet - TLS Protocol JIS-2535 - Add logging of negotiated SSL protocol and cipher suite used for SSL connections 67
JIS-2536 / SI-523570 - New vulnerability found with Jetty JIS-2537 - Upgrade embedded Jetty to version 9.4.53 68
JIS-2548 / SI-551097 - Penetration Scan has Identified Predictable Session Tokens Vulnerability JIS-2569 - Upgrade Session IDs to 128 bits using UUID 68
JIS-2550 / SI-552350 - Issue with Keep Alive JIS-2562 - Add application property setting to OS level TCPKeepAlive 68
JIS-2580 / SI-564462 - Need help getting admin tool working on the Linux servers. JIS-2595 - Modify the method for Deserialization 68
JIS-2581 / SI-564791 - Need Explanation of how ports work JIS-2584 - Improve logging for Xhtml Client Redirection Proxy JIS-2587 - Adding support for enabling the SupportHTTPSOnly setting in HTTP section for Redirection Proxy . . . 68
JIS-2602 / SI-569157 - JIS 9.2.4 beta release attachment JIS-2610 - Update batch files for unbundled JVM or JDK 69
XHTML Client 69
JIS-2346 / PI-5452198 - Action box not reflecting String resources localisation files JIS-2374 - Include radio button text when doing translation for localization . . . 69
JIS-2357 / PI-5457849 - Blackduck Code scan JIS-2361 - CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page JIS-2363 - Remove dependence on URL query string for JIS session information . . 55 JIS-2375 / SI-452878 - Hotfix 3216 and Session closed message JIS-2380 - KeepAlive requests are failing on chrome when language is unset 69
JIS-2377 / SI-452781 - Content-Type charset setting JIS-2424 - Add JIS server property setting to allow configuring Content-Type header charset 69
JIS-2396 / SI-462140 - Session timeout message after 60 seconds sitting idle JIS-2401 - Fix parsing of URL query string for JBS parameter 70

JIS-2416 / SI-468651 - Uncaught type error in Jacada.js and prototype.js JIS-2425 - Fix uncaught TypeError in prototype.js . . . 70
JIS-2517 / SI-510751 - Infinite Loop problem SI449095 came back with latest hotfix JIS-2518 - Fix infinite loop problem for invalid dates . . . 70
JIS-2548 / SI-551097 - Penetration Scan has Identified Predictable Session Tokens Vulnerability JIS-2569 - Upgrade Session IDs to 128 bits using UUID . . . 70
JIS-2581 / SI-564791 - Need Explanation of how ports work JIS-2584 - Improve logging for Xhtml Client Redirection Proxy JIS-2587 - Adding support for enabling the SupportHTTPSOnly setting in HTTP section for Redirection Proxy . . . 70
Limitations of JIS 9.2.4 . . . 71
JIS-2413 / SI-466468 - ContentSecurityPolicy vulnerability . . . 71

JIS Interface Server 9.2.3 Release Notes . . . 72
Installation & Upgrade Information . . . 72
Supported Platforms . . . 72
Recommended Configurations . . . 72
ACE . . . 73
Clients . . . 73
Java Client Applets . . . 73
Java Web Start Clients . . . 73
XHTML Clients . . . 74
Standalone Server . . . 74
Installation and Upgrade Instructions . . . 75
Installing JIS Interface Server 9.2.3 . . . 75
Pack JIS application from previous version of JIS Interface Server . . . 75
Unpack the Jpack in JIS Interface Server 9.2.3 . . . 76
Recompile the JIS Interface Server Application . . . 76
Recompile any Java extensions . . . 77
Create a Runtime Installation . . . 77
Install the JIS Runtime installation . . . 78
Test the Runtime . . . 78
General Data Protection Regulation . . . 78
Retirement of Product Components . . . 79
Java Client Applet Support . . . 79
New Features Included in JIS 9.2.3 . . . 79
OpenJDK Support . . . 79
Open Web Start Support . . . 80
Starting Java Clients With Java Web Start / Open Web Start . . . 80
Starting the JIS Adminstrator With Java Web Start . . . 80
Command Line Import and Export . . . 80
Running ACE in Automatic Mode . . . 81
Changes in the Behavior of ACE Operations . . . 82
Changes in Export . . . 82
Changes in Import . . . 82
Changes in Pack . . . 82
Changes in Unpack . . . 82
Logging . . . 84
Error Handling . . . 84

Limitations 85
Upgrade to Jetty 9.4.40 86
Java Server Page Support in the Embedded Jetty Server 86
JSP Examples 87
Proxy host and port settings for the MakeJar Time Stamp Authority 89
SSL Cipher suite and protocol settings for Java client ports 89
HTTP Request and response buffer size settings 90
HTTP header settings 90
Cache-Control 91
Content-Security-Policy 91
Strict-Transport-Security 91
X-Content-Type-Options 92
X-Frame-Options 92
X-Xss-Protection 92
Disable HTTP methods 93
Secure HTTP Cookies 93
Pause at the end of jacc.bat 93
Detailed Description of Fixes Included in JIS 9.2.3 94
ACE 95
JIS-2163 / 1358933 - Need help with [MakeJar] section of the jacadasv.ini JIS-2164 - Add support for proxy host and port properties to the makejar section of the jacadasv.ini file 95
Java client 96
JIS-2029 / 1292251 - 9.2.2 - Compile Error after upgrading JIS-2033 - Application JNLP file will not run with jar files signed with any other certificate than the SAG certificate JIS-2044 - Add support for enabling JSP processing to embedded Jetty server JIS-2114 - Fix parameter substitution broken by JIS-2044 96
JIS-2240 / 1414399 - Run Inside the browser is not working correctly JIS-2242 - Modify base.html template to make applet frame the size of the browser window . . 96
XHTML Client 96
JIS-2059 / 5304250 - The X and Y position of controls are different from v9.2 and v9.2.2 JIS-2080 - Fix for backward compatibility of checkbox location with JIS 9.2 & 9.2.1 . 83 JIS-2282 / 5427697 - Various GUI controls are not functioning in Chrome JIS-2321 - Fix Xhmtl client date control problem on IE 97
JIS-2283 / 1432003 - Prototype.js receiving error in chrome 97
Installation 97
JIS-2244 - Update signing certificate for 2020-2022 97
Server 97
JIS-2047 / 1302354 - Starting Server Error: java.lang.NoClassDefFoundError: jnlp/ sample/servlet/JnlpDownloadServlet JIS-2048 - Include JacadaFiles/utils/jnlp in JIS XHTML installations 97
JIS-2072 / 5309216 - javax.servlet.UnavailableException: Init parameter 'proxyTo' is required 98
JIS-2082 / 5312165 - Compile in JIS 9.2.2 not including the proper path names resulting in an HTTP Error 404 JIS-2084 - Enable path compaction in JIS Jetty 98
JIS-2090 / 5317186 - Case related to Incidents 5296922 and 5307096 98

JIS-2106 / 5325425 - Not able to clone CheckBox in our application (From 9.2.0 to 9.2.2 Upgrade) JIS-2108 - Not able to clone CheckBox in our application (From 9.2.0 to 9.2.2 Upgrade) JIS-2132 - Fix the names of hidden subcomponents 98
JIS-2123 / 5329638 - JIS 9.2.2 not showing input and output colors correctly JIS-2133 - "Prevent replacement of ""-"" with "" -"" in server output." 98
JIS-2130 / 5334808 - Space before Dash removed in multiple field line comboBox or window title (post fix 5325425) JIS-2131 - "Fix for window title and combo-box entries with "" -"" in them getting changed to ""-"""" 99
JIS-2136 / 5339002 - 2 combo boxes working in JIS 9.2.1 but not JIS 9.2.2 99
JIS-2156 / 5351852 - Problem with JavaClientSSLEnabled 99
JIS-2182 / 5365887 - Fortify scan issues JIS-2188 - Fixes for fortify scan issues . . . 99
JIS-2252 / 5416624 - Disable HTTP OPTIONS Method JIS-2264 - Add property setting to allow disabling HTTP methods 99
JIS-2260 / 5423256 - JIS jetty scan QID 11827 HTTP Security Header not detected JIS-2331 - Add config options for HTTP response headers 99
JIS-2261 / 5423985 - Lack of Response Headers: HSTS JIS-2331 - Add config options for HTTP response headers 100
JIS-2266 / 1428350 - Timing for transactions JIS-2267 - Add logging to make timing of HTTP requests easier 100
JIS-2281 / 5428809 - BBI Blackduck scanning-license and security issues JIS-2329 - Upgrade embedded Jetty to Jetty-9.4.40 100
JIS-2285 - New Vulnerability Items from Client JIS-2309 - Upgrade embedded Jetty to Jetty-9.4.38 100
JIS-2310 / 5439014 - JIS Crashed on AMVC BBI screen JIS-2311 - Fix NullPointerException 100
JIS-2324 / 1445798 - X-Content-Type-Options header for JIS JIS-2331 - Add config options for HTTP response headers 100
JIS-2325 / 1445800 - Security Policy Header JIS-2331 - Add config options for HTTP response headers 101
JIS-2326 / 1445799 - X_XSS Protection Response Header JIS-2331 - Add config options for HTTP response headers 101
Limitations of JIS 9.2.3 101
JIS-2046 / 5298807 - Java clients started via Java Web Start can’t find ports file . . 101
JIS-2157 / 1356981 - JIS Administrator not showing ini settings 102
JIS-2218 / 5389111 - POST Data in Worker Processes 102
JIS-2221 / 1237366 - Add support for server / host SSL protocol and cipher suite settings 89 JIS-2238 / 1298595 - IzPack deployment does not update an existing jacadasv.ini when deploying multiple applications to the same server 102
JIS-2296 / 1362003 - Fix missing table row buttons 102
JIS-2297 / 1347703 - Fix problem with modified JIS application JNLP files 103
JIS-2262 - Session ID in the clear 103

JIS Interface Server 9.2.2 Release Notes 104
Installation & Upgrade Information 104
Supported Platforms 104
Recommended Configurations 104
ACE 105
Clients 105

Java Clients 105
XHTML Clients 105
Standalone Server 106
Application Server Deployment 107
OS400 Components 107
Upgrade Instructions 107
Runtime Installation Upgrade Instructions 107
Retirement of Product Components 108
Java Client Applet Support 108
New Features Included in JIS 9.2.2 108
Java 8 Support 108
Upgrade to IzPack 5.0 108
Java Web Start Support 109
Starting Java Clients With Java Web Start 109
Starting the JIS Adminstrator With Java Web Start 109
Upgrade to Jetty 9.4.2 110
Automatic HTTP to HTTPS Redirection 110
Clickjacking Protection 110
Optional Caching of Table CSS Styesheet Resources 111
Enable Compatibility Mode in Internet Explorer 11 Automatically 111
Detailed Description of Fixes Included in JIS 9.2.2 112
ACE 115
JIS-1655 / 5148126 - Using Wise Installer to Create Runtime Installation JIS-1656 - Wise installer not launched by runtime installation wizard 115
Java client 115
JIS-1626 / 5143850 - Screen Flickering Issue for later 1.7.0_xx JRE versions JIS-1631 - Prevent load balancer from caching Java client HTTP requests 115
JIS-1613 / 5140988 - MessageBox from Client, title: Server not responding JIS-1632 - Fix focus related Java client deadlock 116
JIS-1634 - Java Client Help-->About dialog contains garbage char 116
JIS-1648 / 5147214 - Problem with Keyboard Buffering JIS-1649 - Keyboard buffering enters an infinite loop 116
JIS-1651 / 5147560 - Server Disconnect on Text Entry screen JIS-1652 - Problem sending data when a java.awt.TextArea is the focused control. 103 JIS-1687 / 5155831 - Status of Java 8 certification JIS-1691 - Java 8 certification 116
JIS-1737 - Java 8 support 116
JIS-1746 / 5166996 - Print margin defaults changed with JIS 9.2.1 JIS-1748 - Specify default page margins for print gui and print host screen 117
JIS-1827 - Add FOCUS filter to track the Java client focus behavior 117
JIS-1863 / 1107543 - Issue with scroll bars on tables JIS-1870 - Subapplication specific screen system trigger methods are not invoked. 104 JIS-1942 - Java Web Start support JIS-1950 - Web Start support 117
JIS-1966 / 5244901 - Printing Issue JIS-1968 - Improve printing of image buttons 118
JIS-1993 - Load the JIS administrator as a Java Web Start application 118
XHTML Client 118
JIS-1607 / 5140355 - Delta - Excess Server Logs JIS-1627 - Eliminate Session Dump when calling TotalExit 118
JIS-1650 - KeepAlive requests are not sent by the client 118

JIS-1661 - XSLT and JavaScript refactoring 118
JIS-1671 / 5153539 - Maps Displayed After Natural Terminal Non Conversational Mode Activated JIS-1694 - Implement page loading indicator 119
JIS-1682 / 5155150 - Upgrading to v9.2.1 - Javascript error JIS-1683 - Prototype.js causing JavaScript error with IE8 119
JIS-1693 / 5150037 - Date control issue JIS-1711 - Date control fixes 119
JIS-1701 - Compress response to client 119
JIS-1727 / 5162398 - Application causing delay and loading on JITGUI JIS-1744 - Fix emulator logic related to screen changes 120
JIS-1754 / 5170343 - Delta - Need to inject <!DOCTYPE html> at the beginning of our page JIS-1763 - Add support for the HTML5 DOCTYPE and remove XML declaration 107 JIS-1760 / 5170962 - JIS application does not execute the prompting when prompt is issued from a table field JIS-1761 - Prompt button inside a table does not work 120
JIS-1778 / 5175923 - JIS 9.2.1 XHTML - View Host Screen JIS-1783 - Remove code duplication between view host and printing buttons 121
JIS-1780 / 5175421 - Radio Group issues JIS-1782 - Fix radio button duplicate label on IE 121
JIS-1800 / 5180922 - Look of JacadaList is Corrupted in IE11 JIS-1835 - Adapt table and folded table to HTML5 rendering 121
JIS-1803 / 5182228 - JIS 9.2.1 getPostDate() API JIS-1804 - Change the name attribute from txt to txtarea for text area in non-IE browsers 108 JIS-1810 / 5183374 - Slow transactions when connected to secure host port JIS-1814 - Specify TcpNoDelay when connecting to SSL server 121
JIS-1821 / 5184220 - Behavior In IE11 different concerning the display of the page when submitting it JIS-1866 - Switching to IE11 compatibility mode automatically 122
JIS-1826 - Fix JITGUI on IE11 with HTML5 Doctype 122
JIS-1848 / 5192870 - GEM - Missing XHTMLCSS Stylesheet Resource File - Causing "Expanding Tables" JIS-1852 - Table CSS refresh problem 122
JIS-1895 / 5211802 - Converting from java aplets to XHTML - Norwegian characters JIS-1907 - Use UTF8 encoding when converting the static XML file to a DOM . . 109 JIS-1912 / 5218237 - Clickjacking protection JIS-1915 - Add support for the X-Frame-Options HTTP header to prevent clickjacking 109
JIS-1938 - Change click event behavior for date picker 123
Installation 123
JIS-1639 - Update code signing certificate 123
JIS-1770 - Support for runtime installation Windows uninstall panel 123
JIS-1904 - Update runtime installation to IzPack 5.0 123
JIS-1953 - 2016 code signing certificate 123
Server 123
JIS-1607 / 5140355 - Delta - Excess Server Logs JIS-1627 - Eliminate Session Dump when calling TotalExit 123
JIS-1635 - Administrator runtime settings XHTML section contains empty checkbox . . 111 JIS-1651 / 5147560 - Server Disconnect on Text Entry screen
JIS-1652 - Problem sending data when a java.awt.TextArea is the focused control. 111 JIS-1669 - Upgrade to Jetty 9 124
JIS-1678 - Use Servlet 3.0 AsyncContext in XHTML 124

JIS-1682 / 5155150 - Upgrading to v9.21.- Javascript error JIS-1683 - Prototype.js causing JavaScript error with IE8 124
JIS-1687 / 5155831 - Status of Java 8 certification JIS-1691 - Java 8 certification . 124
JIS-1689 - Use AsyncContext in HTTP proxy servlet 124
JIS-1690 / 5155863 - Error popup during runtime - Couldn't create window
JIS-1692 - Dynamically allocate components per subapplication 125
JIS-1704 / 1093683 - Password/hidden fields being logged. JIS-1707 - Password information printed to server log 125
JIS-1708 / 5158088 - JIS service status on JIS server JIS-1740 - Do something about the missing -Xrs setting 125
JIS-1719 / 5161179 - Formats.ini file not working after ugrade of JIS software from 9.2 to 9.2.1 JIS-1726 - Fix dictionary upper case and separator symbols 125
JIS-1721 / 5162246 - POODLE Security Vulnerability Breaks SSLv3 Secure Browsing JIS-1730 - Fix poodle security vulnerability 125
JIS-1727 / 5162398 - Application causing delay and loading on JITGUI JIS-1744 - Fix emulator logic related to screen changes 126
JIS-1737 - Java 8 support 126
JIS-1753 - HTTP to HTTPS redirection 126
JIS-1802 / 5180721 - Many to One Screen issue JIS-1808 - Many to One - do not collect messages from other dependents 126
JIS-1803 / 5182228 - JIS 9.2.1 getPostDate() API JIS-1804 - Change the name attribute from txt to txtarea for text area in non-IE browsers 114 JIS-1811 / 5185330 - IP redirect JIS-1815 - BindIPAddressForHTTPClient setting is broken . 127
JIS-1810 / 5183374 - Slow transactions when connected to secure host port
JIS-1814 - Specify TcpNoDelay when connecting to SSL server 127
JIS-1829 / 5189161 - Jetty URL case senitivity and ETag support JIS-1838 - Enable Etag response header to improve caching 127
JIS-1846 / 5192404 - Package process for JIS - target root directory coming out in all caps JIS-1847 - Application Server deployment - fix case of RootDir parameter 128
JIS-1863 / 1107543 - Issue with scroll bars on tables JIS-1870 - Subapplication specific screen system trigger methods are not invoked. 115 JIS-1888 / 5210202 - Unable to start application using an obfuscated JettyKeyStore password JIS-1893 - SSL socket cannot read obfuscated keystore password 128
JIS-1895 / 5211802 - Converting from java aplets to XHTML - norwegian characters JIS-1907 - Use UTF8 encoding when converting the static XML file to a DOM . . 115 JIS-1912 / 5218237 - Clickjacking protection JIS-1915 - Add support for the X-Frame-Options HTTP header to prevent clickjacking 115
JIS-1966 / 5244901 - Printing Issue JIS-1968 - Improve printing of image buttons 129
JIS-1973 / 5248661 - Multiple JIS clients continually freeze JIS-1974 - Reduce PS items log verbosity 129
JIS-1977 / 5251062 - Jetty web server configuration question JIS-1978 - Disbable unused HTTP methods and protect server identity 129
JIS-1993 - Load the JIS administrator as a Java Web Start application 129
JIS-1994 - Wize runtime installation - JIS server failed to start 129
Limitations of JIS 9.2.2 129
JIS-1670 - Cannot connect to emulator to capture screens JIS-1679 - Java based screen capture tool 130

JIS-1647 - Software Flow Control JIS-1700 - Typing data using a driver license scanner . . . 130
JIS-1697 - Specific Customizations JIS-1713 - Remember default runtime installation folder . . . 130
JIS-1738 - Unsigned Java client cannot connect to server using SSL . . . 130
JIS-1793 - Project specific configuration for page loading indicator . . . 131
JIS-1849 - JIS server performance issue JIS-1851 - Increase the number of Jetty threads automatically . . . 131
JIS-1830 - DB Bug - Screen JIS-1877 - Delete subapplication do not delete the screen image from the database 118 JIS-2000 - Administrator - Runtime Configuration Tab -Can't open the "Category" combo 118 JIS Interface Server 9.2.1 Release Notes . . . 132
Installation & Upgrade Information . . . 132
Supported Platforms . . . 132
Recommended Configurations . . . 132
ACE . . . 133
Clients . . . 133
Standalone Server . . . 135
Application Server Deployment . . . 136
OS400 Components . . . 136
Upgrade Instructions . . . 136
Retirement of Product Components . . . 137
New Features in Version 9.2.1 . . . 137
Runtime Installation . . . 138
Creating the IzPack runtime installation . . . 139
Creating an installation kit . . . 142
Uninstalling a runtime installation . . . 143
Enhanced Java Client Packaging Procedure . . . 143
Running the Java client in the development environment . . . 144
Packaging application resources into Jar files . . . 144
Parameter Substitution in HTML Launcher Page . . . 145
Pass Through Gateway . . . 148
View Host Screen in XHTML . . . 149
Simplify Redirection Proxy Configuration . . . 150
Secure Single Signon Implementation using Redirection Proxy . . . 151
Proxy Servlet to Server Direct Communication . . . 152
Support Underline and Strike-through Font Attributes . . . 153
Solaris KSSL . . . 154
Jetty HTTP Thread-pool Monitoring . . . 154
Server Business Logic . . . 154
IE10 & IE11 . . . 155
XHTML client . . . 155
Java client . . . 156
Server Remote Management . . . 156
Remote Debugging . . . 157
Activating remote debugging . . . 158
BIN2XML Utility . . . 158
Mobile Demo Web Site . . . 158

Logging Improvements 159
Detailed Description of JIS 9.2.1 Fixes 159
Server 159
#1071275 / JIS-1469 / JIS-1484 - Invoke UserDestroyApplication on unexpected disconnect 159
#5077748 / JIS-1175 / JIS-1389 - WriteUserVariable & GetUserVariables - resulting in null 159
#5094537 / JIS-1285 / JIS-1331 - Running the server on Oracle Linux 160
JIS-1360 - JISAdminServlet requires entering user credentials twice 160
#5085737 / JIS-1224 / JIS-1375 - Security mismatch for password 160
#5108408 / JIS-1393 / JIS-1397 - "Unexpected Host Screen" occurs when using "RemainInScreen: True" 160
JIS-1461 - Data switched between sessions when typing to the host 160
#5115354 / JIS-1457 / JIS-1471 - Can't see all node in Admin console 160
JIS-1473 - Add shutdown hook when server console is disabled 161
#5129821 / JIS-1551 / JIS-1561 - Directory Traversal Security problem 161
JIS-1322 - Application Verifier fixes 161
#5099911 / JIS-1334 / JIS-1337 - Message handling problem 161
#5099911 / JIS-1334 / JIS-1338 – Toolbar subapplication 161
#5124921 / JIS-1526 / JIS-1538 - fix infinite loop in screen interpreter 161
#5103216 / JIS-1358 / JIS-1359 - Checkbox translation failed when using empty settings 149 #5125057 / JIS-1528 / JIS-1530 - ArrayIndexOutOfBoundsException on sendAIDKeysAndWait 162
JIS-1477 - Root node information is not updated in Jam 162
JIS-1570 - passing data between server and administrator 162
#1080114 / JIS-1553 / JIS-1558 - production server not connectable 162
#1072189 / JIS-1480 / JIS-1488 - Problem when the first session does not complete its initialization 162
#5121179 / JIS-1504 / JIS-1598 - Textbox max length set when JIS DoMethod SetVarValueByName used 163
Java Client 163
#5095377 / JIS-1378 / JIS-1380 - printed host screen sometimes lose data 163
#5080361 / JIS-1193 / JIS-1398 - Closing Applet 163
#5110432 / JIS-1416 / JIS-1437 - Rollover image does not work properly on an image button 163
#5115348 / JIS-1455 / JIS-1470 – Screen hanging 164
#5138380 / JIS-1595 / JIS-1599 – Infinite refresh loop 164
JIS-1614 – Simplistic fix for a permission problem introduced by Java 8 164
JIS-1600 – Java 1.7.0_55/1.8.0_05 fix for socket communication 164
XHTML 165
#5118397 / JIS-1486 / JIS-1487 JavaScript permission problem 165
#5122932 / JIS-1516 / JIS-1524 Timeout Redirect 165
ACE 165
#5101893 / JIS-1349 / JIS-1350 Java Exception in Generate Runtime 165
#5107008 / JIS-1383 / JIS-1385 - runtime generation difference between gui and remote 152 #5106317 / JIS-1377 / JIS-1414 - Brown background color in color table changed to Yellow 166
#5111725 / JIS-1420 / JIS-1429 - Deprecated Method used in ACE 166

Limitations of JIS 9.2.1 166
JIS-1594 - XHTML - Combobox - IE11 &IE10 opened above the field 166

JIS Interface Server 9.2 Release Notes 167
Installation & Upgrade Information 167
Supported Platforms 167
Recommended Configurations 167
ACE 168
Clients 168
JIS Standalone Server 169
J2EE Deployment 170
OS400 Components 170
Retirement of Product Components 170
New Features Included in JIS 9.2 171
Running the Server using 64-bit Java 171
Application Verifier 173
Simplified Web Application Deployment 175
Mobile Device Support 176
Simplifying the use of the LoadTest Applet 177
Support for Keyboard buffering 178
Upgrading to Jetty 8.1.5 180
Localization of Checkbox Values 180
Logging and Monitoring Improvements 180
Optimizing the Mechanism which Searches and Identifies Popup Host Screens 181
Client Specific Device Name Assignment 182
Emulator Trace Utilities Integrated into JIS 184
Detailed Description of Fixes Included in JIS 9.2 184
Server 184
JIS-1263: DBCS related infrastructure changes: 184
SI-5073798 (JIS-1156) 184
SI-5076069 (JIS-1170) 185
SI-5069196 (JIS-1122) 185
SI-5074062 (JIS-1155) 185
JIS-1166 185
JIS-1167 185
SI-1057170 (JIS-1247) 185
SI-5073514, SI-5073513 (JIS-1290) 186
Java client 186
SI-5070655 (JIS-1126) 186
SI-5070122 (JIS-1138) 186
SI-5076000 (JIS-1189) 186
SI-5084914 (JIS-1233) 186
SI-1048440 (JIS-1237) 186
SI-5086213 (JIS-1248) 187
SI-5071805 (JIS-1206) 187
SI-5091782 (JIS-1286) 187
XHTML Client 187
SI-5085238 (JIS-1243) 187

JIS-1222 188
Limitations of JIS 9.2 188
SI-5072258 (JIS-1135) 188
SI-5077748 (JIS-1187) 188
SI-5080361 (JIS-1197) 188
JIS-1260 188
JIS 9.2 Appendix: Optimizing the Server for 64-bit Java Version 189

JIS Interface Server 9.1.2 Release Notes 191
Installation & Upgrade Information 191
Supported Platforms 191
Recommended Configurations 191
ACE 192
Clients 192
JIS Standalone Server 193
J2EE Deployment 194
OS400 Components 194
Retirement of the Innovator and Studio Components 195
New Features Included in JIS 9.1.2 195
Creating screen images from Natural Maps 195
Importing Natural maps 196
Creating Popup Windows from Natural Maps 196
Handling Function (F) Keys 198
Natural Maps and JITGUI 201
Simplified HTTPS/SSL Configuration 201
Improving the Keystore Configuration 201
Setting up HTTPS Communication between the XHTML Client and the Server . . . 203
Setting up HTTPS Communication between the Java Client and the Server 203
Setting up SSL Connection between the Java Client and the Server 203
SSL connection between the server and the host 205
IPv6 Support 206
Specifying a Folder where the Java Client Log File will be Saved 206
Logging Messages Improvements 207
Proxy Servlet Improvements 207
Updated JIS Perl to Version 5.12.2.0 207
Session Dump Improvements 208
Access Log 208
Pattern Matching according to Character Attributes 209
Detailed Description of Fixes Included in JIS 9.1.2 211
Installation 211
SI-1033305 211
Java client 211
JIS-647 211
JIS-640 211
SI-5042482 212
SI-5055856 212
JIS-628 212
SI-5041270 212

SI-5059124 212
SI-5061479 212
SI-5056228 212
SI-5058523 213
XHTML Client 213
SI-5057726 213
JIS-599 213
JIS-1043 213
JIS-1038 213
SI-1033559 213
JIS-1025 214
JIS-610 214
JIS-676 214
SI-501580 214
SI-1044586 214
JIS-1102 214
SI-1034309 214
Server 215
SI-5037488 215
SI-5036973 215
JIS-629 215
SI-5061285 215
JIS-1075 215
Innovator 215
SI-5015378 215
Limitations of JIS 9.1.2 216

JIS Interface Server 9.1.1 Release Notes 217
Installation & Upgrade Information 217
Supported Platforms 217
Recommended Configurations 217
ACE 218
Clients 218
JIS Standalone Server 219
J2EE Deployment 220
OS400 Components 220
New Features Included in JIS 9.1.1 221
XHTML 221
Server Changes 221
Monitoring Improvements 222
New Process Attributes 222
Best Practices 223
New System Status Attributes 224
Localization Improvements 225
ACE 225
Runtime Installation 225
Java Client Improvements 226
GUI Printing improvements 226

Detailed Description of Fixes Included in JIS 9.1.1 226
Server 226
AS/400 226
Java client 227
XHTML Client 227
ACE 227
Innovator 227
Limitations of JIS 9.1.1 228

JIS Interface Server 9.1 Release Notes 230
Installation & Upgrade Information 230
Supported Platforms 230
Recommended Configurations 230
ACE 231
Clients 231
Standalone Server 232
J2EE Deployment 232
JIS Java Proxy Servlet 233
OS400 Components 234
Retirement of Product Components 234
New Features Included in JIS 9.1 234
WebSphere 7 Support 234
Usability Improvements in the "Generate Runtime" and "Run Application" Wizards .
. 225 Run Application Wizard Improvements 236
Additional improvements: 236
Changing Default Settings 237
Maintaining Backward Compatibility 237
Restarting the JIS Server 238
JMX Support 239
Connecting to the server using a JMX client application 241
XHTML Page Size Optimization Improvements 242
Server Log 242
Java Client Log 243
XHTML JavaScript Client Log 244
Keyboard shortcut for Java client Print GUI 244
Localization Improvements 245
Localizing Dynamic Control Strings 245
Multiplying Default Control Size by a Pre-Defined Factor 248
Detailed Description of Fixes Included in JIS 9.1 250
Installation 250
Server 250
Java client 250
XHTML Client 251
AS/400 251
Innovator 252
Limitations of JIS 9.1 252
JIS 9.1 Appendix: JMX Code Samples 253

JIS Interface Server 9.0.4 Release Notes 258
Installation & Upgrade Information 258
Supported Platforms 258
Recommended Configurations 258
ACE 259
Clients 259
JIS Standalone Server 260
J2EE Deployment 260
JIS Java Proxy Servlet 261
AS400 Components 261
Installing JIS 9.0.4 server on OS400 V6R1 262
Preparing your OS400 V6R1 machine Java environment 262
Preparing for the runtime installation 262
Retirement of Product Components 263
New Features Included in JIS 9.0.4 263
Logging Improvements 263
Simplifying JIS Windows Service Configuration 264
Using JAM as an Applet in JIS Standalone Server 264
Secure Login to JISAdminServlet 265
JIS Administrator Command Line Operations 266
Modifications made to the J2EE Deployment Procedure (XHTML only) 267
Upgrade to Jetty 6.1 267
Backward compatibility 268
Running the JacadaProxyServlet as part of the JIS Server 268
Reduction of the size of the XHTML file 269
Allowing the User to Adjust the Java Client Debug Level 270
Post Class Path 270
New Methods for Handling User Variables 270
Rebranding 272
Detailed Description of Fixes Included in JIS 9.0.4 272
Server 272
Java client 273
XHTML Client 273
ACE 274
Limitations of JIS 9.0.4 274

JIS Interface Server 9.0.3 Release Notes 277
Installation & Upgrade Information 277
Supported Platforms 277
Recommended Configurations 277
ACE 277
Clients 278
Standalone Server 278
J2EE Deployment 279
JIS Java Proxy Servlet 280
AS400 Components 280
Installation & Upgrade Instructions 280
AS400 Components Installation 281

JIS Common Installation for J2EE Deployment 281
JIS Runtime Installation for Proprietary Server 281
License 282
Usage of Dongle 282
New Features Included in JIS 9.0.3 282
Changes in the Product Name 282
Runtime Installation Improvements 282
Changes in the installation process 283
Simplifying the Printing Emulation Configuration (XHTML) 284
Improved Host Language Support 285
Introduction 285
Using this Feature 285
Customization 286
Log Files 287
Backwards Compatibilility 287
Parameters 288
Parameters for Backwards Compatible Settings (to be used only when not using the "Descriptor" mechanism) 289
Supported Languages 289
Java Client "About" Dialog Box 294
"Host Print Transform" Printing using Java Services 295
Exposing the XHTML Page DOM for Java Extensions 295
Enforcing Strong Encryption 298
AutoSkip Supported in XHTML 298
Detailed Description of Fixes Included in JIS 9.0.3 299
Security 299
Server 299
Java client 299
XHTML Client 300
ACE 301
Innovator 301
Administration Console 301
Emulation 302
Limitations of JIS 9.0.3 302

JIS Interface Server 9.0B Release Notes 303
New Features Included in JIS 9.0B 303
Command-Line Access to ACE 303
Running ACE in Automatic Mode 304
Changes in the Behavior of ACE Operations 304
Changes in Pack 304
Changes in Unpack 304
Logging 306
Error Handling 306
Limitations 307
Improved User Interface 308
Introduction 308
Mouse rollover mode 309

Mouse wheel support 309
Arrow buttons and scroll bars 309
Rounded rectangles for group boxes and frames 309
Calendar control 309
Client persistent storage 309
Table column reordering 310
Table row sorting 310
Known Limitations 310
Additional Enhancements 311
Accessing the application server's HTTP session from an XHTML extension (ATL- 27368) 311
Loading resources from an XHTML extension (ATL-27368) 311
XHTML Date control enhancements (EU-05423) 312
Displaying digits in Java Client spin box controls (ATL-28026) 313
Support for monochrome terminals (ATL-28319) 313
SSL Connection (ATL-29082) 313
Changing XHTML Message Boxes (ATL-27839) 314
Menus in XHTML (ATL-28670) 314
JIS 9.0B Appendix: Using Secured Telnet in JIS V9.0B 315
Settings and Short Names 315
JacadaRootDir 315
JRELibDir 315
Important Files 315
CACERTS 315
Secure Telnet between JIS and the Mainframe 315
Methodology 315

JIS Interface Server 9.0A07 Release Notes 318
New Features Included in JIS 9.0A07 318
Refreshing the XHTML Client When a Page on the Host is Updated 318

JIS Interface Server 9.0A06 Release Notes 319
New Features Included in JIS 9.0A06 319
Platform Support 319
Support for Keyboard Buffering 319
Activating Keyboard Buffering 319
Known Limitations 321

JIS Interface Server 9.0A05 Release Notes 322
New Features Included in JIS 9.0A05 322
API available to trigger server methods 322
Printing 322
Support for portrait and landscape printing 322
Enabling specifying margins 323
Enabling specifying the paper size 323
Improved handling of underlines in print output 323
Enabling printing to any printer 323

New printer parameters 323

JIS Interface Server 9.0A02 Release Notes 324
New Features Included in JIS 9.0A02 324
Enabling reconnecting to a database after the connection or session fails 324
Limiting the size of the server log files 325

JIS Interface Server 9.0A01 Release Notes 326
New Features Included in JIS 9.0A01 326
Enable opening a window in a maximized state 326
Deploying a service to a J2EE Server 327
In XHTML java extensions 327
In javascript extensions 327
In the XHTML html extensions 327

JIS Interface Server 9.0A00 Release Notes 328
New Features Included in JIS 9.0A00 328
XINIT keyword in BMS maps now supported 328
Maximum permitted size of ACE method increased 328
Print setup dialog can be skipped 328
New methods for setting colors of selected cells 329

List of Figures

JIS 9.2.1 Application Blocked by Security Settings 134
JIS 9.2.1 Control Panel Security Tab 134
JIS 9.2.1 Java Security Warning Dialog Box 135
JIS 9.2.1 Create Runtime Installation Wizard - Select Installer Type Dialog Box 139
JIS 9.2.1 Create Runtime Installer Console Window 140
JIS 9.2.1 IzPack Welcome Dialog Box 141
JIS 9.2.1 IzPack Uninstaller Dialog Box 143
JIS 9.2 Create Runtime Wizard - Select Web Application Root Directory Dialog Box . . . 175
JIS 9.1.2 Create Runtime Images Wizard - Select Screen Images Type Dialog Box 196
JIS 9.1.2 NatFKeys 200
JIS 9.1.2 NatDynamicFKeys 200
JIS 9.1.2 Pattern Definitions View Character Attributes Dialog Box 210
JIS 9.1.1 New Process Attributes 222
JIS 9.1.1 New system status attributes 223
JIS 9.1 WebSphere 7 JVM Settings 235
JIS 9.1 Application Wizard 236
JIS 9.1 Restarting the JIS server via the server console 238
JIS 9.1 Restarting the JIS server via JAM’s command line operations 238
JIS 9.1 Restarting the JIS server via a JMX monitoring tool 239
JIS 9.1 Connecting via Client 241
JIS 9.1 Window Options Control Sizing 249
JIS 9.0.3 Transferring to Solaris 283
JIS 9.0.3 Application Configuration 284

List of Tables

JIS 9.3 Included component updates 15
JIS 9.3 New and changed settings 16
JIS 9.2.5 Java Web Start Client Supported Browser and Java Versions 19
JIS 9.2.5 XHTML Client Supported Operating Systems and Browsers 19
JIS 9.2.5 XHTML Client Supported Runtime Environments 20
JIS 9.2.5 Standalone Server Supported Environments 20
JIS 9.2.5 MakeJar Server Configuration Propeties 29
JIS 9.2.4 Java Web Start Client Supported Browser and Java Versions 36
JIS 9.2.4 XHTML Client Supported Operating Systems and Browsers 36
JIS 9.2.4 XHTML Client Supported Runtime Environments 37
JIS 9.2.4 Standalone Server Supported Environments 38
JIS 9.2.4 New MakeJar Server Configuration Propeties 48
JIS 9.2.3 Java Applet Client Supported Browser and Java Versions 73
JIS 9.2.3 Java Web Start Client Supported Browser and Java Versions 74
JIS 9.2.3 XHTML Client Supported Operating Systems and Browsers 74
JIS 9.2.3 Standalone Server Supported Environments 75
JIS 9.2.3 Command-Line ACE Error Resolution 84
JIS 9.2.2 Java Client Supported Browser and Java Versions 105
JIS 9.2.2 XHTML Client Supported Operating Systems and Browsers 106
JIS 9.2.2 Standalone Server Supported Environments 106
JIS 9.2.2 XHTML Client Supported Runtime Environments 107
JIS 9.2.2 X-Frame-Options Directives 111
JIS 9.2.1 Java Client Supported Browser and Java Versions 133
JIS 9.2.1 XHTML Client Supported Operating Systems and Browsers 135
JIS 9.2.1 Standalone Server Supported Operating Systems 135
JIS 9.2.1 XHTML Client Supported Environments 136
JIS 9.2.1 Client HTML Context Values 146
JIS 9.2 Java Client Supported Browser and Java Versions 168
JIS 9.2 XHTML Client Supported Operating Systems and Browsers 169
JIS 9.2 Standalone Server Supported Operating Systems 169
JIS 9.2 XHTML Client Supported Environments 170
JIS 9.2 Java Client Keyboard Buffering Parameters 178
JIS 9.1.2 Java Client Supported Browser and Java Versions 192
JIS 9.1.2 XHTML Client Supported Operating Systems and Browsers 193
JIS 9.1.2 Standalone Server Supported Environments 193
JIS 9.1.2 XHTML Client Supported Runtime Environments 194
JIS 9.1.2 Natural Parser Window Properties 197
JIS 9.1.1 Java Client Supported Browser and Java Versions 218
JIS 9.1.1 XHTML Client Supported Operating Systems and Browsers 219
JIS 9.1.1 Standalone Server Supported Environments 219
JIS 9.1.1 XHTML Client Supported Runtime Environments 220
JIS 9.1 Java Client Supported Browser and Java Versions 231
JIS 9.1 XHTML Client Supported Operating Systems and Browsers 232
JIS 9.1 Standalone Server Supported Environments 232

JIS 9.1 XHTML Client Supported Runtime Environments 233
JIS 9.1 Java Proxy Servlet Supported Environments 233
JIS 9.0.4 Java Client Supported Browser and Java Versions 259
JIS 9.0.4 XHTML Client Supported Operating Systems and Browsers 259
JIS 9.0.4 Standalone Server Supported Environments 260
JIS 9.0.4 XHTML Client Supported Runtime Environments 260
JIS 9.0.4 Java Proxy Servlet Supported Environments 261
JIS 9.0.4 JAM Status Command Return Codes 266
JIS 9.0.3 Java Client Supported Browser and Java Versions 278
JIS 9.0.3 XHTML Client Supported Operating Systems and Browsers 278
JIS 9.0.3 Standalone Server Supported Environments 279
JIS 9.0.3 XHTML Client Supported Runtime Environments 279
JIS 9.0.3 Java Proxy Servlet Supported Environments 280
JIS 9.0.3 Supported Languages 290
JIS 9.0.3 Language Descriptor to Codepage Mapping 291
JIS 9.0B Command-Line ACE Error Resolution 306
JIS 9.0A06 Parameters for Keyboard Buffereing 320

List of Examples

JIS 9.3 HTTP configuration settings 9
JIS 9.3 Content Security Policy settings 9
JIS 9.2.5 GUiSys TN3270 Property Examples 33
JIS 9.2.5 GUiSys TN5250 Property Examples 33
JIS 9.2.4 GUiSys TN3270 Property Examples 53
JIS 9.2.4 GUiSys TN5250 Property Examples 53
JIS 9.2.3 Sample buildapp.xml 83
JIS 9.2.2 HTTP Redirection Properties 110
JIS 9.2.2 XFrameOptions Properties 110
JIS 9.2.2 CSS Stylesheet Table Caching Properties 111
JIS 9.2.1 Makejar Jar Signing Properties: 145
JIS 9.2.1 Setting the Value of the DebugLevel Applet Parameter 147
JIS 9.2.1 Setting JIS GatewayMappings Properties 148
JIS 9.2 Application Verifier IncludePanelFiles Setting 174
JIS 9.2 Localization of Checkbox Values 180
JIS 9.2 DeviceNameTemplate 183
JIS 9.1.2 RedirectionPolicy Server Address Setting 206
JIS 9.1.2 Java Client Log Setting 206
JIS 9.1.2 Jetty NCSA Access Log 208
JIS 9.1.1 Recommended resource file encodings 225
JIS 9.1 Running The Application with Firefox 237
JIS 9.1 Running the Application with Internet Explorer 237
JIS 9.1 Server Log Options 242
JIS 9.1 Customizing the Default Print Keyboard Shortcut 244
JIS 9.1 Localizing Control Strings by SubApplication and Prefix 246
JIS 9.1 Localizing Control Strings by Prefix 247
JIS 9.1 Changing an application's host 253
JIS 9.1 Stopping the server 254
JIS 9.1 Gathering data regarding running sessions 256
JIS 9.0.4 Generating An Encrypted Password 265
JIS 9.0.4 PostClasspath Setting 270
JIS 9.0.4 Handling User Variables From Within an ACE Method 271
JIS 9.0.4 Handling User Variables From Within a Server Side Extension 271
JIS 9.0.3 Extending Product Language Descriptors 294
JIS 9.0.3 Using the XHTML Page DOM in a Java Extension 296
JIS 9.0B Sample buildapp.xml 305
JIS 9.0B Loading Resources From an XHTML Extension 311
JIS 9.0B CSTSSLSocketFactory.java 316
JIS 9.0A00 Setting Cell Background Color 329

Overview

About JIS Interface Server

Welcome to the Release Notes for JIS Interface Server, an automated development architecture that generates Java and XHTML clients for enterprise applications. It uses JIS Interface Server's Automated Conversion Environment to convert character-based host screens into feature rich graphical clients in 100% Java source code or XHTML pages. Consequently, mainframe and iSeries applications can be delivered through an intranet, or over the Internet using the generated client interface.

Organization

The *JIS Interface Server: Release Notes* are organized as follows:

- "JIS Interface Server 9.3 Release Notes" on page 3
- "JIS Interface Server 9.2.5 Release Notes" on page 18
- "JIS Interface Server 9.2.4 Release Notes" on page 35
- "JIS Interface Server 9.2.3 Release Notes" on page 72
- "JIS Interface Server 9.2.2 Release Notes" on page 104
- "JIS Interface Server 9.2.1 Release Notes" on page 132
- "JIS Interface Server 9.2 Release Notes" on page 167
- "JIS Interface Server 9.1.2 Release Notes" on page 191
- "JIS Interface Server 9.1.1 Release Notes" on page 217
- "JIS Interface Server 9.1 Release Notes" on page 230
- "JIS Interface Server 9.0.4 Release Notes" on page 258
- "JIS Interface Server 9.0.3 Release Notes" on page 277
- "JIS Interface Server 9.0B Release Notes" on page 303
- "JIS Interface Server 9.0A07 Release Notes" on page 318
- "JIS Interface Server 9.0A06 Release Notes" on page 319
- "JIS Interface Server 9.0A05 Release Notes" on page 322
- "JIS Interface Server 9.0A02 Release Notes" on page 324
- "JIS Interface Server 9.0A01 Release Notes" on page 326
- "JIS Interface Server 9.0A00 Release Notes" on page 328

Note: Release notes for prior releases of JIS are provided for historical purposes only. The current release notes should be referenced for current software and hardware requirements.

JIS Interface Server 9.3 Release Notes

About this release

JIS 9.3 modernizes the web and Java runtime stack, replaces the active Borland Database Engine (BDE) application database with SQLite, adopts the Software AG signed-license format, and introduces substantial web, native-code, and build-artifact security hardening.

The release also improves embedded-server startup and HTTPS behavior, migration of existing applications and libraries, Java Web Start codebase handling, and screen generation on Windows 11 24H2.

Release highlights

- Modernized embedded web runtime based on Jetty 12.0.35 and EE10 components.
- Updated Tomcat, JSP, and EL integration to Jakarta APIs and Tomcat 11.0.24 components.
- Automatic transformation of generated user Java sources from `javax.servlet` to `jakarta.servlet` before compilation.
- SQLite application database with automatic migration of existing BDE-based applications and libraries.
- Harmonized signed Software AG XML licensing across installed, runtime, WAR, standalone, Wise, and InstallAnywhere packages.
- Enhanced Content Security Policy support, including report-only operation and per-response nonce authentication.
- Authenticated redirect-URL protection using persistent AES-GCM keys.
- Modernized the JIS Perl toolchain from Strawberry Perl 5.12.2.0 to 5.42.3.1 for improved compatibility and security.
- Broad hardening of native C and C++ code and Perl scripts based on static-analysis findings.
- Improved Windows 11 24H2 GAL screen generation and nested-window positioning.
- Expanded digital-signing coverage and verification for distributed executables, libraries, and JAR files.

Installation and upgrade information

JIS 9.3 requires Java 21 or newer and is built with JDK 21 instead of JDK 17.

Migrating BDE application databases to SQLite

JIS 9.3 uses SQLite as the active application database. During application access and package unpacking, JIS detects eligible legacy BDE databases and starts the migration utility when a corresponding SQLite database is not present.

The migration covers application and library data, subapplications, layouts, jump tables, external applications, binary fields, and other persisted application metadata. The source BDE data is retained after a successful conversion to provide a recovery path.

JIS no longer installs BDE into the machine-wide Borland Shared BDE directory under Windows Common Files. A private copy of the legacy BDE components is included only under the JIS migration area so that existing databases can be converted.

Licensing

A new license key is required to install JIS 9.3.

Please fill out the following form https://empower.softwareag.com/Products/OrderProducts/default.aspx.

Contact Logistics at Generic.LogisticsServiceCenterGER@softwareag.com to obtain the new key. The license will be delivered in an `*.xml` format.

JIS 9.3 uses a signed Software AG XML license. Installation and repair allow the administrator to browse for and validate the license using the selected JDK. The selected license is stored as `SAGlicense.xml`, and `license.dat` points to that file.

The same license payload is included consistently in runtime installations, WAR deployments, standalone bundles, Wise packages, and InstallAnywhere packages. For ACE Designer compatibility, the installer derives a legacy-compatible CD key from supported feature flags in the signed license and writes it to the existing RTCP configuration.

When replacing a license through JAM, the file chooser remembers the last selected folder. Expiration and grace-period reminders have also been improved.

Installation and Upgrade Instructions

The installation process for JIS 9.3 differs from earlier versions. The license key must be obtained from Logistics before attempting to install JIS. See the licensing instructions under Installation and Upgrade Information.

JDK 21 must be installed prior to running the JIS 9.3 installation.

The following steps describe the procedure for installing JIS Interface Server 9.3 and upgrading from a previous version:

- Pack JIS Applications from Previous Version of JIS Interface Server
- Install JIS Interface Server 9.3
- Configure JIS Interface Server 9.3 to Run on an External Java Installation
- Unpack the Jpack in JIS Interface Server 9.3
- Recompile the JIS Interface Server Application
- Recompile any Java Extensions
- Create a Runtime Installation:
  - Java client
  - XHTML client
- Install the JIS Runtime Installation
- Test the Runtime

Pack JIS Applications from Previous Version of JIS Interface Server

The following steps describe the procedure for packing a JIS application from the previous version:

- Open the application in the ACE Designer in the previous version of JIS
- Select Utility/Pack from the menu
- Follow the prompts in the pack wizard:
  - Select all libraries and sub applications
  - Select Single File
  - Select Java Files
  - Select all checkboxes for DDS, SDF, and Screens
  - Uncheck Temporary Auxiliary Files and Installations
  - Select Don’t pack any configuration files
  - Include any extra image files that are not found in the `<Application>images` folder
  - Include any additional files being used by the application that do not default to JIS. Examples are 3rd party software, jar files, etc.
- Allow the Pack Process to complete

Install JIS Interface Server 9.3

The JIS installer must be run with Administrator Rights.

- Install JDK 21
- Launch the JIS Interface Server InstallShield Wizard.
- Install JIS 9.3 in a new directory. Do not install over the top of a previous version of JIS.
- Select the installed JDK folder as described below.
- Select the new XML license file obtained from Logistics when prompted by the installer.

Configure JIS Interface Server 9.3 to Run on an External Java Installation

During installation, select the JDK folder by specifying the path up to the Java directory.

For Example: `C:\Program Files\Zulu\zulu-21`

Unpack the Jpack in JIS Interface Server 9.3

- Open the JIS ACE Designer
- Select Utility/Unpack
- Follow the prompts in the Unpack Wizard
  - Select the Jpack that was created from the previous version of JIS by clicking on the Browse button
  - Select everything under items to unpack. By default everything will be selected
  - Select Packed Application
  - Compare Library Names to make sure the original and new library names match
  - Select java files
  - Select DDS, SDF, Screens from Input Directories
  - Select Runtime Files
  - Select Don't Unpack Any Configuration files
  - Select … to point any 3rd party files and images into the new location
- Click finish

When the application starts unpacking there will be a message that appears about converting the database files. This message is expected.

Recompile the JIS Interface Server Application

The following steps describe the procedure for compiling a JIS application:

- Open the JIS ACE Designer
- Select File/Open Application and open the application
- Select File/Generate Runtime and follow the prompts in the wizard
  - Select Runtime Type Java, XHTML or both
  - Select Server Platform Windows or Linux
  - Select Libraries to include
  - Select all libraries to recompile
  - Select all sub applications to include in runtime
  - Select all sub applications to process
  - Add your host name or IP address and port number
- Click Finish

Recompile any Java Extensions

- Open the Windows File Explorer and select `<JIS9.3>\jacadafiles\src\appls\<Application>`
- In the `<JIS9.3>\jacadafiles\src\appls\<Application>\user` folder double click the `jacc.bat` file
- In the `<JIS9.3>\jacadafiles\src\appls\<Application>\server\user` folder double click the `jacc.bat` file
- Repeat the previous two steps for each library `<JIS9.3>\jacadafiles\src\appls\<Library>`

Create a Runtime Installation

Java client

The following steps describe the procedure for creating a runtime installation for a JIS Java client application:

- Open the `jacadasv.ini` file from the previous version of JIS in the `<JIS>\jacadafiles\classes` folder
- Copy the `[MakeJar]` section
- Open the `<JIS9.3>\jacadafiles\classes\jacadasv.ini` file and paste the `[MakeJar]` section into the file and save it.
- Open the application in the JIS ACE Designer

- Create the runtime installation:
  - Select Utility/Create Runtime Installation
  - Select Stand Alone Server for the default JIS server or Application server for the J2EE deployment
  - Create IzPack Installation
  - Select Runtime type
  - Select Full Runtime for the scope
  - Select Server Platform
  - Select the Default Installation Directory
  - Select any image files for the installation splash screen
  - Click Finish

XHTML client

The following steps describe the procedure for creating a runtime installation for a JIS XHTML client application:

- Open the application in the JIS ACE Designer
- Create the runtime installation:
  - Select Utility/Create Runtime Installation
  - Select Stand Alone Server for the default JIS server or Application server for the J2EE deployment
  - Create IzPack Installation
  - Select Runtime type
  - Select Full Runtime for the scope
  - Select Server Platform
  - Select the Default Installation Directory
  - Select any image files for the installation splash screen
  - Click Finish

Install the JIS Runtime Installation

The following steps describe the procedure for installing a JIS application runtime:

- Copy the contents of the `<JIS9.3>\appls\<Application>\install` folder to the server where the JIS runtime installation will be installed.
- Install the Azul server Java 21 runtime

- Install the JIS server runtime:
  - For Windows, run the following command:
    `RuntimeInstallation.exe`
  - For Linux, use the following command to install the Runtime Installation:
    `java -jar RuntimeInstallation.jar`

Test the Runtime

Perform the necessary steps to insure the integrity of the upgraded application.

Configuration and encoding

The default XML output encoding is now UTF-8 through the `XmlEncoding=UTF-8` installer and build setting.

Clustered installations that accept the same encrypted redirect URLs use the same protected URL-encryption key.

Clients should add the following settings to the `[HTTP]` section of the `jacadasv.ini` file only if these values are not already configured:

Example 1. JIS 9.3 HTTP configuration settings

```
[HTTP]
KeystorePassword=JettyKeyStore
JAMUsername=(username)
JAMPassword=(password)
```

To enable Content Security Policy (CSP), clients should add the following settings to the `[HTTP]` section of the `jacadasv.ini` file:

Example 2. JIS 9.3 Content Security Policy settings

```
[HTTP]
CSPAuthenticate=1
ContentSecurityPolicy="default-src 'none'; connect-src 'self';
script-src 'self' 'nonce'; style-src 'self' 'nonce'; img-src 'self'
data:; font-src 'self' https://fonts.googleapis.com
https://fonts.gstatic.com data:; block-all-mixed-content; frame-src
'self'; frame-ancestors 'self';"
```

General Data Protection Regulation

While JIS Interface Server does not collect or store any user personal information directly, the server may log personal information from the host application depending on the specific host application and server debug level.

At debug level 70 or higher, client access logging will be enabled and server / host interaction will also be logged. These logs may contain personal information depending on the host application and server transactions.

Setting the server debug level to 10 or lower will disable access logging between the client and the server and will also disable logging of server / host interaction.

Any personal information in the JIS Interface Server logs can be removed by deleting the server logs.

New features and enhancements

Jakarta and embedded web-runtime modernization

The embedded web runtime has been updated from Jetty 11.0.25 to Jetty 12.0.35 with EE10 servlet, web application, proxy, and session components. Tomcat integration now uses Tomcat 11.0.24 APIs and Jasper components, Jakarta Server Pages 4.0.0, and Jakarta Expression Language 6.0.1.

JIS-generated Java sources that reference `javax.servlet` are transformed to `jakarta.servlet` by Eclipse Transformer before compilation. This provides a migration path for generated applications while the runtime moves to the Jakarta namespace.

Perl toolchain modernization

The JIS build and packaging toolchain now uses Strawberry Perl 5.42.3.1 instead of the older 5.12.2.0 distribution. This upgrade modernizes the Perl runtime used by JIS automation, improves compatibility with current Windows environments, strengthens argument validation and file-handling practices, and keeps the product aligned with a supported Perl version for ongoing maintenance and security hardening.

The update also refreshes JIS scripts and helper modules to reflect modern Perl behavior, reducing compatibility issues in build, packaging, and deployment automation while preserving the existing JIS operational workflows.

SQLite application database and legacy migration

The active application database is now `dbm.sqlite.db`, accessed through SQLite rather than a system-wide BDE installation. Migration can run automatically when an old database is detected and also participates in package unpacking and library import flows.

Migration handling has been strengthened for packed applications, JPACK libraries, strict layouts, subapplication library names, binary buffers, and interrupted or partial conversions.

Software AG license harmonization

License installation, validation, storage, repair, and packaging now use the common signed Software AG XML-license mechanism. The license is propagated through all supported JIS deployment-package types, including WAR output.

Enhanced Content Security Policy support

The existing `ContentSecurityPolicy` capability has been extended with two controls:

- `CSPReportOnly` allows administrators to evaluate a policy by emitting the report-only response header before enforcing it.
- `CSPAuthenticate` enables per-response CSP nonces and moves generated inline scripts, styles, and event handlers into session-specific protected resources where required.

Generated CSP resources are isolated by session and cleaned up when sessions end and when the server starts. CSP headers are also applied consistently to generated error responses.

Protected redirect URLs

Redirect-URL tokens now use authenticated AES-GCM encryption with a versioned, persistent key. The implementation uses a unique initialization vector for every token and rejects modified, malformed, legacy, or oversized values.

The optional `UrlEncryptionKeyFile` setting selects the persistent key file. An explicitly configured key must use an absolute path, must be outside the HTTP resource base, and must be protected as a server secret. If no explicit path is configured, JIS uses its runtime classes area.

Security and hardening

JIS 9.3 includes the following defense-in-depth improvements:

- Per-response CSP nonces and report-only policy evaluation.
- Extraction and session isolation of generated inline JavaScript, CSS, and event-handler content when authenticated CSP is enabled.
- Authenticated redirect-URL encryption with tamper detection and persistent key management.
- Rejection of profile-file path traversal, invalid nested paths, absolute-path escapes, and symbolic-link escapes.
- Protection of sensitive resources, including `UrlEncryption.key`, `WEB-INF`, and `META-INF` content.
- HTML escaping of substituted HTTP header, parameter, and cookie values.
- Generic proxy and error responses that avoid exposing exception classes, local paths, backend details, or sensitive configuration data.
- Safer configuration loading: unreadable or missing required configuration causes an explicit failure, and a failed reload does not replace the last valid configuration.
- Additional validation for protected contexts, resource mappings, and security-related configuration.
- Native C and C++ bounds, index, allocation, integer-overflow, parsing, and buffer-handling checks.
- Safer Perl argument validation, three-argument file opening, and reduced shell interpretation for child-process invocation.
- Expanded signing and verification of distributed executable, library, and Java artifacts.

Detailed description of fixes

Embedded server and HTTPS

- Corrected JSP servlet registration and embedded-server HTTPS configuration, including SNI behavior.
- Improved Jetty startup sequencing so dependent coordinators and dispatchers are started in the required order.
- Improved session-allocation retry behavior during startup and reduced timing-related initialization failures.

- Made keystore initialization synchronous and added clearer startup and error handling.
- Updated gzip, servlet constraint, logging, and error-processing behavior for Jetty 12.
- Increased the default server-status wait period to reduce false startup failures on slower systems.

Java, JRO, and Java Web Start

- Corrected JRO and JNLP codebase resolution so application-server deployments do not incorrectly append an `admin` path segment.
- Added fallback to the configured JAM URL when a separate codebase is not available.
- Corrected null path-information handling in the Jakarta and Tomcat request flow.
- Repaired JDK selection and reselection behavior during install and repair.

Application database and packaging

- Corrected migration and unpacking of packed applications and application libraries.
- Corrected strict-layout joins and write-back behavior that could otherwise corrupt migrated layout information.
- Corrected handling of subapplication library names, external applications, jump-table data, binary fields, and large buffers.
- Prevented automatic migration from hanging in affected unpacking flows.
- Improved migration logging and error handling while preserving the original BDE data for recovery.
- Removed obsolete migration libraries after the converter packaging was consolidated.
- Corrected longstanding issues that previously failed silently and were uncovered while implementing and testing the database-migration feature.

Licensing

- Corrected license selection, validation, and copying during new installation and repair.
- Corrected `license.dat` generation so it consistently references `SAGlicense.xml`.
- Corrected inclusion and lookup of the license in WAR, standalone, Wise, InstallAnywhere, and static installer manifests.

- Added generation of a legacy-compatible CD key from supported signed-license feature flags for ACE Designer and stored it in the existing RTCP configuration.
- Improved expiration and grace-period reminders.

Screen generation and browser behavior

- Corrected GAL row and column detection on Windows 11 24H2.
- Corrected relative offsets when generating layouts for nested windows.
- Removed legacy CSS layout compensation from the default generated-CSS path because it caused incorrect layouts for newly generated stylesheets.

Build, signing, and diagnostics

- Corrected JAR-signing argument construction when a separate keystore store password is not configured.
- Reduced excessive UUID and session-map diagnostic logging.
- Improved consistency between generated product-version data and packaged classes.
- Expanded signing coverage and certificate-chain and timestamp verification for delivered artifacts.

Included component updates

Table 2 - 1 : JIS 9.3 Included component updates

Component	JIS 9.2.5	JIS 9.3
Eclipse Jetty	11.0.25	12.0.35
Apache Tomcat API	Unversioned packaged API	11.0.24
Apache Tomcat Embed Jasper	11.0.8	11.0.24
Apache Tomcat Jasper EL	11.0.8	11.0.24
JSP API	Javax JSP 2.3.0	Jakarta Server Pages 4.0.0
Expression Language API	Javax EL 3.0.0	Jakarta EL 6.0.1
Eclipse Java compiler	3.42.0	3.46.0
Apache Commons CLI	1.2	1.9.0
Eclipse Transformer CLI	Not included	1.0.0
Strawberry Perl	5.12.2.0	5.42.3.1
Software AG licensing library	Not included in this form	SAGLIC 5.7.9.0

New and changed settings

Table 2 - 2 : JIS 9.3 New and changed settings

Area	Setting	Default	Description
HTTP	`ContentSecurityPolicy`	Existing installation value	Existing CSP policy setting; nonce handling is enhanced when CSP authentication is enabled.
HTTP	`CSPReportOnly`	`false`	Sends `Content-Security-Policy-Report-Only` instead of the enforcing header.
HTTP	`CSPAuthenticate`	`false`	Enables per-response nonces and protected session-specific resources for generated inline content.
Redirection proxy	`UrlEncryptionKeyFile`	Runtime classes area	Selects the persistent AES-GCM key file used for protected redirect URLs.
Installer and build	`XmlEncoding`	`UTF-8`	Selects the encoding used for generated XML.

Compatibility and behavior changes

- The servlet, JSP, and EL stack now uses Jakarta namespaces. The standard JIS generation path transforms generated `javax.servlet` references before compilation.
- BDE is no longer installed system-wide or used as the active application database. A private BDE payload remains available for legacy database conversion.
- XML generation now defaults to UTF-8.
- Authenticated CSP changes the handling of generated inline script, style, and event-handler content. Report-only operation is available separately.
- All nodes that need to consume the same protected redirect URL must share the same protected key material.

JIS Interface Server 9.2.5 Release Notes

Installation & Upgrade Information

Supported Platforms

JIS Interface Server 9.2.5 is supported on the following platforms:

- Windows 10 Professional, Ultimate and Enterprise Edition (64-bit)
- Windows 11 Professional, Ultimate and Enterprise Edition (64-bit)
- Windows Server 2016 Standard and Enterprise (64-bit)
- Windows Server 2019 Standard and Enterprise (64-bit)
- Windows Server 2022 Standard and Enterprise (64-bit)
- Red Hat Enterprise Linux 8 for x86 (64-bit)
- Red Hat Enterprise Linux 9 for x86 (64-bit)

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for Java versions, browser versions and application server versions supported by their respective vendors. Being so, when a vendor no longer supports a Java version, browser version or application server version, Software GmbH will discontinue supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported Java, browser or application server version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows 10

- Windows 11
- Windows Server 2019
- Windows Server 2022

Clients

Java Web Start Clients

The Java Web Start Client has been tested on the following operating systems, browser and Java versions:

- Windows 10
- Windows 11
- Windows Server 2019
- Windows Server 2022

Table 3 - 3 : JIS 9.2.5 Java Web Start Client Supported Browser and Java Versions

Browser	JRE
Chrome	Oracle Java 17, Azul Zulu Java 17
Edge	Oracle Java 17, Azul Zulu Java 17
Firefox	Oracle Java 17, Azul Zulu Java 17

XHTML Clients

The XHTML client has been tested with the following operating system and browser versions:

Table 3 - 4 : JIS 9.2.5 XHTML Client Supported Operating Systems and Browsers

Operating System	Browser
Windows 10	Chrome, Edge, Firefox
Windows 11	Chrome, Edge, Firefox

Table 3 - 4 : JIS 9.2.5 XHTML Client Supported Operating Systems and Browsers

Operating System	Browser
Windows Server 2019	Chrome, Edge, Firefox
Windows Server 2022	Chrome, Edge, Firefox
Red Hat Linux 8	Firefox
Red Hat Linux 9	Firefox

Application Server Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 3 - 5 : JIS 9.2.5 XHTML Client Supported Runtime Environments

Application Server or Web Container	Java Runtime Environment	Operating System
OpenLiberty 24	Azul Zulu Java 17	Windows 11
Tomcat 10.0	Azul Zulu Java 17	Windows 11, Red Hat 9

Standalone Server

The JIS standalone server has been tested in the following environments:

Table 3 - 6 : JIS 9.2.5 Standalone Server Supported Environments

Operating System	Java Version
Windows 10	Oracle Java 17, Azul Zulu Java 17
Windows 11	Oracle Java 17, Azul Zulu Java 17

Table 3 - 6 : JIS 9.2.5 Standalone Server Supported Environments

Operating System	Java Version
Windows Server 2019	Oracle Java 17, Azul Zulu Java 17
Windows Server 2022	Azul Zulu Java 17
Red Hat Linux 8	Oracle Java 17, Azul Zulu Java 17
Red Hat Linux 9	Oracle Java 17, Azul Zulu Java 17

Installation and Upgrade Instructions

The installation process for JIS 9.2.5 differs from earlier versions. Unlike JIS 9.2.4, which was distributed as a ZIP file for extraction, JIS 9.2.5 is distributed only as an executable (.exe) installer. A ZIP file installation option is not available for this release.

When you run the installer, you will be prompted with two options:

- Install a new instance of the application.
- Maintain and upgrade an existing installation.

The following steps describe the procedure for installing JIS Interface Server 9.2.5 and upgrading from a previous version:

- JIS Interface Server 9.2.5 Release Notes
- Pack JIS Applications from Previous Version of JIS Interface Server
- Install JIS Interface Server 9.2.5
- Configure JIS Interface Server 9.2.5 to Run on an External Java Installation
- Unpack the Jpack in JIS Interface Server 9.2.5
- Recompile the JIS Interface Server Application
- Recompile any Java Extensions
- Create a Runtime Installation:
  - Java client
  - XHTML client
- Install the JIS Runtime Installation
- Test the Runtime

Note: When installing JIS 9.2.5 on Windows, the JIS installer must be run with administrator permissions.

Pack JIS Applications from Previous Version of JIS Interface Server

The following steps describe the procedure for packing a JIS application from the previous version:

- Open the application in the ACE Designer in the previous version of JIS
- Select Utility/Pack from the menu
- Follow the prompts in the pack wizard:
  - Select all libraries and sub applications
  - Select Single File
  - Select Java Files
  - Select all checkboxes for DDS, SDF, and Screens
  - Uncheck Temporary Auxiliary Files and Installations
  - Select Don't pack any configuration files
  - Include any extra image files that are not found in the <Application>images folder
  - Include any additional files being used by the application that do not default to JIS. Examples are 3rd party software, jar files, etc.
- Allow the Pack Process to complete

Install JIS Interface Server 9.2.5

- Copy the CD key from the RTCP section of the `ace.ini` or `ace400.ini` file in root directory of the previous JIS installation: `<JIS>\ace.ini` for mainframe installations or `<JIS>\ace400.ini` for AS/400 installations:

  ```
  [RTCP]
  CDKey=<JIS_CDKey>
  ```
- Install JIS 9.2.5 in a new directory. Do not install over the top of a previous version of JIS.
- Install Azul OpenJDK Java 17
- Launch the JIS Interface Server InstallShield Wizard.
- Select the JDK folder by specifying the path up to the Java 17 directory.

Configure JIS Interface Server 9.2.5 to Run on an External Java Installation

The below step is required when running JIS 9.2.5 on an external Java installation.

- Launch the JIS Interface Server InstallShield Wizard.
- Select the JDK folder by specifying the path up to the Java 17 directory.

Unpack the Jpack in JIS Interface Server 9.2.5

- Open the JIS ACE Designer
- Select Utility/Unpack
- Follow the prompts in the Unpack Wizard
  - Select the Jpack that was created from the previous version of JIS by
  - clicking on the Browse button
  - Select everything under items to unpack. By default everything will be selected
  - Select Packed Application
  - Compare Library Names to make sure the original and new library names match
  - Select java files
  - Select DDS, SDF, Screens from Input Directories
  - Select Runtime Files
  - Select Don't Unpack Any Configuration files
  - Select … to point any 3rd party files and images into the new location
- Click finish
- For each application, replace <JIS>\appls\<applName>\install\shortcutSpec.xml with <JIS>\RunTime\shortcutSpec.xml
- Regenerate application run times
- Create new runtime installations
- Install the new runtime installations

Recompile the JIS Interface Server Application

The following steps describe the procedure for compiling a JIS application:

- Open the JIS ACE Designer
- Select File/Open Application and open the application
- Select File/Generate Runtime and follow the prompts in the wizard

  - Select Runtime Type Java, XHTML or both
  - Select Server Platform Windows, Linux, Solaris, AIX
  - Select Libraries to include
  - Select all libraries to recompile
  - Select all sub applications to include in runtime
  - Select all sub applications to process
  - Add your host name or IP address and port number
- Click Finish

Recompile any Java Extensions

- Open the Windows File Explorer and select `<JIS9.2.5>\jacadafiles\src\appls\<Application>`
- In the `<JIS9.2.5>\jacadafiles\src\appls\<Application>\user` folder double click the `jacc.bat` file
- In the `<JIS9.2.5>\jacadafiles\src\appls\<Application>\server\user` folder double click the jacc.bat file
- Repeat the previous two steps for each library `<JIS9.2.5>\jacadafiles\src\appls\<Library>`

Create a Runtime Installation

Java client

The following steps describe the procedure for creating a runtime installation for a JIS Java client application:

- Open the `jacadasv.ini` file from the previous version in the `<JIS>\jacadafiles\classes` folder
- Copy the `[MakeJar]` section
- Open the `<JIS9.2.5>\jacadafiles\classes\jacadasv.ini` file and paste the `[MakeJar]` section into the file and save it.
- Open the application in the JIS ACE Designer
- Create the runtime installation:
  - Select Utility/Create Runtime Installation
  - Select Stand Alone Server for the default JIS server or Application server for the J2EE deployment
  - Create IzPack Installation
  - Select Runtime type
  - Select Full Runtime for the scope

- Select Server Platform
- Select the Default Installation Directory
- Select any image files for the installation splash screen
- Click Finish

XHTML client

The following steps describe the procedure for creating a runtime installation for a JIS XHTML client application:

- Open the application in the JIS ACE Designer
- Create the runtime installation:
  - Select Utility/Create Runtime Installation
  - Select Stand Alone Server for the default JIS server or Application server for the J2EE deployment
  - Create IzPack Installation
  - Select Runtime type
  - Select Full Runtime for the scope
  - Select Server Platform
  - Select the Default Installation Directory
  - Select any image files for the installation splash screen
  - Click Finish

Install the JIS Runtime Installation

The following steps describe the procedure for installing a JIS application runtime:

- Copy the contents of the `<JIS9.2.5>\appls\<Application>\install` folder to the server where the JIS runtime installation will be installed.
- Install the Azul server JRE
- Install the JIS server runtime:
  - For windows, run the following command:

    ```
    RuntimeInstallation.exe
    ```
  - For all other server platforms use the following command to install the Runtime Installation:

    ```
    java -jar RuntimeInstallation.jar
    ```

Test the Runtime

Perform the necessary steps to insure the integrity of the upgraded application.

General Data Protection Regulation

While JIS Interface Server does not collect or store any user personal information directly, the server may log personal information from the host application depending on the specific host application and server debug level.

At debug level 70 or higher, client access logging will be enabled and server / host interaction will also be logged. These logs may contain personal information depending on the host application and server transactions.

Setting the server debug level to 10 or lower will disable access logging between the client and the server and will also disable logging of server / host interaction.

Any personal information in the JIS Interface Server logs can be removed by deleting the server logs.

Retirement of Product Components

As of release 9.2.5 the following product components are retired:

- Java 8 Support
- Tomcat 8 Support

New Features Included in JIS 9.2.5

The following new features have been added for JIS 9.2.5:

- Updated Operating System Support
- Java Version 17 Support
- Azul Java 17 Support
- Tomcat 10 Support

Updated Operating System Support

Expanded compatibility to include Windows Server 2022. This update ensure the product can be seamlessly installed, configured, and run on Windows Server 2022.

Java Version 17 Support

JIS Interface Server now supports Java 17 for both compile-time and run-time operations.

Azul Java 17 Support

JIS Interface Server now supports Azul Java 17, enabling applications to be built and executed using the Azul Zulu distribution of Open JDK 17.

Tomcat 10 Support

JIS Interface Server now supports Apache Tomcat 10 for deploying and running applications.

Server Configuration Properties

Support for the following server configuration properties have been added in JIS 9.2.5:

- [HTTP]ContentTypeCharset
- [HTTP]RelativeRedirectAllowed
- [MakeJar]certchain
- [MakeJar]sigfile
- [MakeJar]signedjar
- [MakeJar]digestalg
- [MakeJar]sigalg
- [MakeJar]tsacert
- [MakeJar]tsapolicyid
- [MakeJar]tsadigestalg
- [MakeJar]altsigner
- [MakeJar]altsignerpath
- [MakeJar]internalsf
- [MakeJar]sectionsonly
- [MakeJar]protected
- [MakeJar]providerName
- [MakeJar]providerClass
- [MakeJar]providerArg
- [MakeJar]strict
- [MakeJar]revCheck
- [MakeJar]addprovider
- [MakeJar]javaOption
- [MakeJar]conf
- [MakeJar]addOptions

HTTP Server Configuration Properties

Support for the following HTTP server configuration properties have been added in JIS 9.2.5:

- ContentTypeCharset
- RelativeRedirectAllowed

ContentTypeCharset

The `ContentTypeCharset` property in the `HTTP` section can be used to configure a charset directive for the HTTP Content-Type header sent in response for static HTML pages. Xhtml client launch pages, for example. If the `ContentTypeCharset` property is set, the value of the property will be used as the character set in the Content-Type header char set directive. For example, give the following `jacadasv.ini` setting:

```
[HTTP]
ContentTypeCharset=utf-16
```

the following Content-Type header will be included in responses for requests for static HTML pages:

```
Content-Type: text/html; charset=utf-16
```

If the `ContentTypeCharset` property is unset, no char set directive will be included in the Content-Type header included in responses for requests for static HTML pages:

```
Content-Type: text/html
```

> **Note:** The default value for the `ContentTypeCharset` property is unset.

RelativeRedirectAllowed

When enabled, `RelativeRedirectAllowed` allows the HTTP server to redirect URLs for server directories which do not end in a slash to be redirected to a relative location. For example:

```
http://<jiServer>:8080/images
```

would be redirected to:

```
/images/
```

This mitigates web poisoning attacks which use the HTTP HOST header to redirect browser requests to a compromised web site.

In addition, a property setting: `RelativeRedirectAllowed` has been added to the HTTP section of the JIS server INI file:

```
[HTTP]
```

```
RelativeRedirectAllowed=1
```

The `RelativeRedirectAllowed` property is enabled by default.

MakeJar Server Configuration Properties

In order to allow more flexibility in the code signing process, the following options have been added to the `MakeJar` section of the server INI file:

Table 3 - 7 : JIS 9.2.5 MakeJar Server Configuration Propeties

Property Name	Description
certchain	Name of alternative certchain file `certchain=<file>`
sigfile	Name of .SF/.DSA file `sigfile=<file>`
signedjar	Name of signed JAR file `signedjar=<file>`
digestalg	Name of digest algorithm `digestalg=<algorithm>`
sigalg	Name of signature algorithm `sigalg=<algorithm>`
tsacert	Public key certificate for Time Stamping Authority `tsacert=<alias>`
tsapolicyid	TSAPolicyID for Time Stamping Authority `tsapolicyid=<oid>`
tsadigestalg	Algorithm of digest data in Time Stamping request `tsadigestalg=<algorithm>`

Table 3 - 7 : JIS 9.2.5 MakeJar Server Configuration Propeties

Property Name	Description
altsigner	Class name of an alternative signing mechanism `altsigner=<class>`
altsignerpath	Location of an alternative signing mechanism `altsignerpath=<pathlist>`
sectionsonly	Don't compute hash of entire manifest `sectionsonly=0`
internalsf	Include the .SF file inside the signature block `internalsf=0`
protected	Keystore has protected authentication path `protected=0`
providerName	Provider name `providerName=<name>`
providerClass	Name of cryptographic service provider's master class file and constructor argument `providerClass=<class>`
providerArg	Optional providerClass constructor argument `providerArg=<arg>`
strict	Treat warnings as errors `strict=0`
revCheck	Enable certificate revocation check `revCheck=0`

Table 3 - 7 : JIS 9.2.5 MakeJar Server Configuration Propeties

Property Name	Description
addprovider	Add security provider by name (e.g. SunPKCS11) `addprovider=<name>`
providerArg	Optional security providerconfigure argument `providerArg<arg>`
javaOption	Specify a JVM option or argument `javaOption=<arg>`
conf	Specify a pre-configured options file `conf=<url>`
addOptions	Specify an additional jarsigner option or argument `addOptions=<arg>`

With the exception of the `javaOption` and `addOptions` properties, these properties directly correspond to jarsigner command line options. For example, the `certchain` property corresponds to the `-certchain` option. For those options that take arguments, the value of the property corresponds to the argument for the option. Using the `-certchain` option as an example, the server INI file setting:

```
[MakeJar]certchain=C:/CERTS/chain.crt
```

would result in the following arguments being appended to the jarsigner command line:

```
-certchain c:\CERTS\chain.crt
```

The `internalsf`, `sectionsonly`, `protected`, `strict` and `revCheck` properties correspond to boolean command line options with values of 1 or 0 indicating the option is enabled or disabled. For example,

```
[MakeJar]strict=1
```

would enable the `-strict` command line option. All of these properties are disabled by default.

The command line options corresponding to the `revCheck`, `addProvider` and `conf` properties are for Java 17.

The `javaOption` property corresponds to the `-J<javaOption>` Java VM command line option and `<javaOption>` is passed through to the Java VM running the jarsigner command.

The `addOptions` property does not correspond to any jarsigner command line options. If set, it is appended to the jarsigner command line.

Application Configuration Properties

Support for the following application configuration properties have been added in JIS 9.2.5:

- EnableTCPKeepAlive
- SupportedProtocols
- SupportedCipherSuites

EnableTCPKeepAlive

The `EnableTCPKeepAlive` property in the `GUISys TN3270` and `GUISys TN5250` sections allow the user to enable the operating system `SO_KEEPALIVE` feature for server/host socket connections.

> **Note:** The `SO_KEEPALIVE` feature is platform dependent and KeepAlive timeouts must be set at the O/S level.

SupportedProtocols and SupportedCipherSuites

The `SupportedProtocols` and `SupportedCipherSuites` properties in the `GUISys TN3270` and `GUISys TN5250` sections of the application INI file can be used to configure the SSL protocols and cipher suites enabled for SSL communication between JIS server and the host.

The value of the `SupportedProtocols` property will be used to set the protocols enabled for communication between the JIS server and the host. If the `SupportedProtocols` property is unset, the default protocols supported by the JVM, excluding SSLv3 and SSLv2Hello will be enabled.

The value of the `SupportedCipherSuites` property will be used to set the cipher suites enabled for communication between the JIS server and the host. If the `SupportedCipherSuites` property is unset, the default cipher suites supported by the JVM will be used.

The default value for the `SupportedProtocols` and `SupportedCipherSuites` properties is unset.

Note: The `SupportedProtocols` and `SupportedCipherSuites` properties in the `GUISys TN3270` and `GUISys TN5250` sections only apply to server / host communication and do not replace or affect the settings of the `SupportedProtocols` or `SupportedCipherSuites` properties client / server communication in the `GeneralParameters` or `HTTP` sections of the server INI file.

TLSv1.3 requires Java 8u261 or later.

Example3. JIS 9.2.5 GUiSys TN3270 Property Examples

```
[GUISys TN3270]
;SecureHostConnection=1
;SupportedProtocols=TLSv1.2,TLSv1.3
;SupportedCipherSuites=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_DHE_DSS
_WITH_AES_128_GCM_SHA256
;EnableTCPKeepAlive=0
```

Example4. JIS 9.2.5 GUiSys TN5250 Property Examples

```
[GUISys TN5250]
;SecureHostConnection=1
;SupportedProtocols=TLSv1.2,TLSv1.3
;SupportedCipherSuites=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_DHE_DSS
_WITH_AES_128_GCM_SHA256
;EnableTCPKeepAlive=0
```

Detailed Description of Fixes Included in JIS 9.2.5

This release brings product upgrades only, and no bug fixes are included.

Limitations of JIS 9.2.5

The following are known limitations in JIS 9.2.5:

- JIS-2413 / SI-466468 - ContentSecurityPolicy vulnerability

JIS-2413 / SI-466468 - ContentSecurityPolicy vulnerability

If ContentSecurityPolicy settings are enabled, policy settings which do not include:

```
script-src 'unsafe-inline' 'unsafe-eval'; style-src 'unsafe-inline
```

will likely cause problems for JIS XHTML client applications.

JIS Interface Server 9.2.4 Release Notes

Installation & Upgrade Information

Supported Platforms

JIS Interface Server 9.2.4 is supported on the following platforms:

- Windows 10 Professional, Ultimate and Enterprise Edition (64-bit)
- Windows 11 Professional, Ultimate and Enterprise Edition (64-bit)
- Windows Server 2016 Standard and Enterprise (64-bit)
- Windows Server 2019 Standard and Enterprise (64-bit)
- Windows Server 2022 Standard and Enterprise (64-bit)
- Red Hat Enterprise Linux 8 for x86 (64-bit)
- Red Hat Enterprise Linux 9 for x86 (64-bit)

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for Java versions, browser versions and application server versions supported by their respective vendors. Being so, when a vendor no longer supports a Java version, browser version or application server version, Software GmbH will discontinue supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported Java, browser or application server version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows 10

- Windows 11
- Windows Server 2019

Clients

Java Web Start Clients

The Java Web Start Client has been tested on the following operating systems, browser and Java versions:

- Windows 10
- Windows 11
- Windows Server 2019
- Windows Server 2022

Table 4 - 8 : JIS 9.2.4 Java Web Start Client Supported Browser and Java Versions

Browser	JRE
Chrome	Oracle Java 1.8.0, 17, Azul Zulu Java 8, 17
Edge	Oracle Java 1.8.0, 17, Azul Zulu Java 8, 17
Firefox	Oracle Java 1.8.0, 17, Azul Zulu Java 8, 17

XHTML Clients

The XHTML client has been tested with the following operating system and browser versions:

Table 4 - 9 : JIS 9.2.4 XHTML Client Supported Operating Systems and Browsers

Operating Syste	Browser
Windows 10	Chrome, Edge, Firefox
Windows 11	Chrome, Edge, Firefox
Windows Server 2019	Chrome, Edge, Firefox

Table 4 - 9 : JIS 9.2.4 XHTML Client Supported Operating Systems and Browsers

Operating Syste	Browser
Windows Server 2022	Chrome, Edge, Firefox
Red Hat Linux 8	Firefox
Red Hat Linux 9	Firefox
iPadOs 14.5.1	Chrome, Safari

Application Server Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 4 - 10 : JIS 9.2.4 XHTML Client Supported Runtime Environments

Application Server or Web Container	Java Runtime Environment	Operating System
OpenLiberty 24	Azul Zulu Java 8 & 17	Windows 11
Tomcat 8.5	Azul Zulu Java 8 & 17	Windows 11
Tomcat 9.0	Azul Zulu Java 8 & 17	Windows 11, Red Hat 9
WebSphere Liberty 21 & 22	Azul Zulu Java 8 & 17	Windows 11

Standalone Server

The JIS standalone server has been tested in the following environments:

Table 4 - 11 : JIS 9.2.4 Standalone Server Supported Environments

Operating System	Java Version
Windows 10	Oracle Java 1.8.0 & 17, Azul Zulu Java 8 &17
Windows 11	Oracle Java 1.8.0 & 17, Azul Zulu Java 8 &17
Windows Server 2019	Oracle Java 1.8.0 & 17, Azul Zulu Java 8 &17
Windows Server 2022	Azul Zulu Java 8
Red Hat Linux AS8	Oracle Java 1.8.0 & 17, Azul Zulu Java 8 &17
Red Hat Linux AS9	Oracle Java 1.8.0 & 17, Azul Zulu Java 8 &17]

Installation and Upgrade Instructions

The installation process for JIS 9.2.4 differs from previous versions in that, it is being distributed as a zip-file to be extracted over an existing installation of JIS 9.2.3.

The following steps describe the procedure for installing JIS Interface Server 9.2.4 and upgrading from a previous version:

- JIS Interface Server 9.2.4 Release Notes
- Install JIS Interface Server 9.2.4
- Configure JIS Interface Server 9.2.4 to Run on an External Java Installation
- Unpack the Jpack in JIS Interface Server 9.2.4
- Recompile the JIS Interface Server Application
- Recompile any Java Extensions
- Create a Runtime Installation:
  - Java client
  - XHTML client
- Install the JIS Runtime Installation
- Test the Runtime

Note: When installing JIS 9.2.4 on Windows, the JIS installer must be run with administrator permissions.

Pack JIS Applications from Previous Version of JIS Interface Server

The following steps describe the procedure for packing a JIS application from a previous version:

- Open the application in the ACE Designer in the previous version of JIS
- Select Utility/Pack from the menu
- Follow the prompts in the pack wizard:
  - Select all libraries and subapplications
  - Select Single File
  - Select Java Files
  - Select all checkboxes for DDS, SDF, and Screens
  - Uncheck Temporary Auxiliary Files and Installations
  - Select Don't pack any configuration files
  - Include any extra image files that are not found in the <Application>images folder
  - Include any additional files being used by the application that are not default to JIS. Examples are 3rd party software, jar files, etc.
- Allow the Pack Process to complete

Install JIS Interface Server 9.2.4

For the inital installation of JIS 9.2.4 it is recommended you start with a fresh install of JIS 9.2.3 named to indicate it is a JIS 9.2.4 installation. The JIS 9.2.4 zip-file will then be extracted over the JIS 9.2.3 installation:

- Copy the CD key from the RTCP section of the `ace.ini` or `ace400.ini` file in the root directory of the previous JIS installation: `<JIS>\ace.ini` for mainframe installations or `<JIS>\ace400.ini` for AS/400 installations:

  ```
  [RTCP]
  CDKey=<JIS_CDKey>
  ```
- Install JIS 9.2.3 in a new directory. Do not install over the top of a previous version of JIS.
- Install the latest hotfix for JIS 9.2.3 on the new JIS 9.2.4 installation
- Remove or rename the `<JIS>\JacadaFiles\utils\IzPack` directory. It will be replaced in the next step.
- Extract the JIS 9.2.4 zip-file over the new JIS 9.2.4 installation

- Edit `<JIS>\ace.ini` and/or `<JIS>\ace400.ini` to update the `[Program]Version` property to: `9.02.0004`
- Remove any `jar.pack.gz` files from the `JacadaFiles\classes\cst` and `JacadaFiles\classes\appls\<appl>\lib` directories in your JIS 9.2.4 development installation directory or `classes` and `classes\cst` directories in your JIS runtime installation directory
- Install Azul OpenJDK Java 8 or 17

Configure JIS Interface Server 9.2.4 to Run on an External Java Installation

These steps are only required when running JIS 9.2.4 on an external Java installation.

- Rename the `<JIS>\JacadaFiles\utils\IzPack` directory.
  This step is not required but will make it easier to find any batch or other files referencing the embedded Java installation.
- Edit the Java compiler command in the Java panel of the `Options/Runtime Generation Options` dialog in JIS ACE Designer, replacing the path to the embedded JRE with the path to your external JDK installation.

> **Note:** Spaces in the Java compiler comand path or arguments must be quoted to be interpretted correctly.

- Edit any batch or other files that reference the embedded JRE, including at a minimum: `jacadasv.bat`, `CreateRuntimeInstallion.bat`, `InvokeRuntimeInstallation.bat` and any `jacc.bat` files required for compiling custom code.

Unpack the Jpack in JIS Interface Server 9.2.4

The following steps describe the procedure for unpacking a JIS application from from a previous version into the JIS 9.2.4 development installation:

- Open the JIS ACE Designer
- Select Utility/Unpack
- Follow the prompts in the Unpack Wizard:
  - Select the jpack that was created from the previous version of JIS by clicking on the Browse button
  - Select everything under items to unpack. By default everything will be selected
  - Select Packed Application

  - Compare Library Names to make sure the original and new library names match
  - Select java files
  - Select DDS, SDF, Screens from Input Directories
  - Select Runtime Files
  - Select Don't Unpack Any Configuration files
  - Select … to point any 3rd party files and images into the new location
- Click finish
- Remove or rename `<JIS>\appls\<applName>install\shortcutSpec.xml`, it will be replaced the next time you create a runtime installation.

Recompile the JIS Interface Server Application

The following steps describe the procedure for compiling a JIS application:

- Open the JIS ACE Designer
- Select File/Open Application and open the application
- Select File/Generate Runtime and follow the prompts in the wizard
  - Select Runtime Type Java, XHTML or both
  - Select Server Platform Windows, Linux, Solaris, AIX
  - Select Libraries to include
  - Select all libraries to recompile
  - Select all subapplications to include in runtime
  - Select all subapplications to process
  - Add your host name or IP address and port number
- Click Finish

Recompile any Java Extensions

- Open the Windows File Explorer and select `<JIS9.2.4>\jacadafiles\src\appls\<Application>`
- In the `<JIS9.2.4>\jacadafiles\src\appls\<Application>\user` folder double click the `jacc.bat` file
- In the `<JIS9.2.4>\jacadafiles\src\appls\<Application>\server\user` folder double click the jacc.bat file
- Repeat the previous two steps for each library `<JIS9.2.4>\jacadafiles\src\appls\<Library>`

Create a Runtime Installation

Java client

The following steps describe the procedure for creating a runtime installation for a JIS Java client application:

- Open the `jacadasv.ini` file from the previous version in the `<JIS>\jacadafiles\classes` folder
- Copy the `[MakeJar]` section
- Open the `<JIS9.2.4>\jacadafiles\classes\jacadasv.ini` file and paste the `[MakeJar]` section into the file and save it.
- Open the application in the JIS ACE Designer
- Create the runtime installation:
  - Select Utility/Create Runtime Installation
  - Select Stand Alone Server for the default JIS server or Application server for the J2EE deployment
  - Create IzPack Installation
  - Select Runtime type
  - Select Full Runtime for the scope
  - Select Server Platform
  - Select the Default Installation Directory
  - Select any image files for the installation splash screen
  - Click Finish

XHTML client

The following steps describe the procedure for creating a runtime installation for a JIS XHTML client application:

- Open the application in the JIS ACE Designer
- Create the runtime installatin:
  - Select Utility/Create Runtime Installation
  - Select Stand Alone Server for the default JIS server or Application server for the J2EE deployment
  - Create IzPack Installation
  - Select Runtime type
  - Select Full Runtime for the scope
  - Select Server Platform
  - Select the Default Installation Directory
  - Select any image files for the installation splash screen

- Click Finish

Install the JIS Runtime Installation

The following steps describe the procedure for installing a JIS application runtime:

- Copy the contents of the `<JIS9.2.4>\appls\<Application>\install` folder to the server where the JIS runtime installation will be installed.
- Install the Azul server JRE
- Install the JIS server runtime:
  - For windows, run the following command:

    ```
    RuntimeInstallation.exe
    ```
  - For all other server platforms use the following command to install the RuntimeInstallation:

    ```
    java -jar RuntimeInstallation.jar
    ```

Test the Runtime

Perform the necessary steps to insure the integrity of the upgraded application.

General Data Protection Regulation

While JIS Interface Server does not collect or store any user personal information directly, the server may log personal information from the host application depending on the specific host application and server debug level.

At debug level 70 or higher, client access logging will be enabled and server / host interaction will also be logged. These logs may contain personal information depending on the host application and server transactions.

Setting the server debug level to 10 or lower will disable access logging between the client and the server and will also disable logging of server / host interaction.

Any personal information in the JIS Interface Server logs can be removed by deleting the server logs.

Retirement of Product Components

As of release 9.2.4 the following product components are retired:

- Java Client Applet Support
- Legacy Unix Support

Java Client Applet Support

With the end of support for Internet Explorer, running JIS Java clients as applets is no longer supported.

Legacy Unix Support

The following Legacy Unix platforms are no longer supported:

- AIX
- HP-UX
- Solaris

New Features Included in JIS 9.2.4

The following new features have been added for JIS 9.2.4:

- HTTP RelativeRedirectAllowed Enabled by Default
- Upgrade to Jetty 9.4.53
- Server/Host SSL Protocol & Cipher Suite Settings
- Content-Type Header Charset Setting
- Run JAM as a JNLP Application
- Jetty Logging Setting
- 128-Bit Session IDs
- OS Level TCPKeepAlive Setting
- XHTML RedirectionProxy with SupportHTTPSOnly
- Jarsigner Command Line Option Settings
- Unbundled JVM or JDK
- IzPack Upgrade to Version 5.2.3

HTTP RelativeRedirectAllowed Enabled by Default

A configuration change has been made to the JIS Jetty HTTP server. The `RelativeRedirectAllowed` setting has been made configurable and is now enabled by default.

See the New HTTP Server Configuration Properties section below for details.

Upgrade to Jetty 9.4.53

The JIS Jetty HTTP server has been upgraded to version 9.4.53 in order to utilize the latest security and performance enhancements.

Server/Host SSL Protocol & Cipher Suite Settings

Two new application configuration options have been added in the `GUISys TN3270` and `GUISys Tn5250` sections of the application INI file: `SupportedProtocols` and `SupportedCipherSuites`. These can be used to configure SSL communication between the JIS server and host.

See the New Application Configuration Properties section below for details.

Content-Type Header Charset Setting

A new server configuration property was added to allow setting the HTTP Content-Type header charset directive for static HTML pages.

See the New Server Configuration Properties section below for details.

Run JAM as a JNLP Application

It is now possible to start JAM as a JNLP application, allowing JAM to be started via JNLP on Open Web Start as well as Java Web Start. Prior to this fix, JAM could only be run via JNLP as an applet which is not supported on Open Web Start.

Jetty Logging Setting

The Jetty log level is now configurable via a properties file. Jetty logging is set to `WARN` by default but can be configured by modifying the `jetty-logging.properties` file included in the `JacadaFiles\classes` directory. For runtime installations, copy the `jetty-logging.properties` file to the `classes` directory in the runtime installation. The Jetty log file is overwritten each time the JIS server is started.

128-Bit Session IDs

To improve security, Java and XHTML client session IDs are now implemented using 128-bit UUIDs.

OS Level TCPKeepAlive Setting

A new application configuration option hasbeen added in the `GUISys TN3270` and `GUISys Tn5250` sections of the application INI file: `EnableTCPKeepAlive`. This setting can be used to enable the operating system SO_KEEPALIVE feature on server/host socket connections.

See the New Application Configuration Properties section below for details.

XHTML RedirectionProxy with SupportHTTPSOnly

The XHTML RedirectionProxy has been enhanced to include support for HTTPS connections between the RedirectionProxy and the JIS server worker processes, allowing the server to be configured with the [HTTP]SupportHTTPSOnly setting enabled in conjunction with the RedirectionProxy.

Jarsigner Command Line Option Settings

A number of new server configuration properties were added to accomodate the new private key storage requirements for code signing certificates.

See the New MakeJar Server Configuration Properties section below for details.

Unbundled JVM or JDK

With the release of JIS 9.2.4, JIS is not longer required to run on the bundled JVM and may be configured to run on an externally installed JVM or JDK.

IzPack Upgrade to Version 5.2.3

The IzPack runtime installation generator included with JIS 9.2.4 has been upgraded to version 5.2.3.

New Server Configuration Properties

Support for the following server configration properties have been added in JIS 9.2.4:

- [HTTP]ContentTypeCharset
- [HTTP]RelativeRedirectAllowed
- [MakeJar]certchain

- [MakeJar]sigfile
- [MakeJar]signedjar
- [MakeJar]digestalg
- [MakeJar]sigalg
- [MakeJar]tsacert
- [MakeJar]tsapolicyid
- [MakeJar]tsadigestalg
- [MakeJar]altsigner
- [MakeJar]altsignerpath
- [MakeJar]internalsf
- [MakeJar]sectionsonly
- [MakeJar]protected
- [MakeJar]providerName
- [MakeJar]providerClass
- [MakeJar]providerArg
- [MakeJar]strict
- [MakeJar]revCheck
- [MakeJar]addprovider
- [MakeJar]javaOption
- [MakeJar]conf
- [MakeJar]addOptions

New HTTP Server Configuration Properties

Support for the following HTTP server configuration properties have been added in JIS 9.2.4:

- ContentTypeCharset
- RelativeRedirectAllowed

ContentTypeCharset

The `ContentTypeCharset` property in the `HTTP` section can be used to configure a charset directive for the HTTP Content-Type header sent in response for static HTML pages. Xhtml client launch pages, for example. If the `ContentTypeCharset` property is set, the value of the property will be used as the character set in the Content-Type header charset directive. For example, give the following `jacadasv.ini` setting:

```
[HTTP]
ContentTypeCharset=utf-16
```

the following Content-Type header will be included in responses for requests for static HTML pages:

```
Content-Type: text/html; charset=utf-16
```

If the `ContentTypeCharset` property is unset, no charset directive will be included in the Content-Type header included in responses for requests for static HTML pages:

```
Content-Type: text/html
```

> **Note:** The default value for the `ContentTypeCharset` property is unset.

RelativeRedirectAllowed

When enabled, `RelativeRedirectAllowed` allows the HTTP server to redirect URLs for server directories which do not end in a slash to be redirected to a relative location. For example:

```
http://<jiServer>:8080/images
```

would be redirected to:

```
/images/
```

This mitigates web poisoning attacks which use the HTTP HOST header to redirect browser requests to a compromised web site.

In addition, a new property setting: `RelativeRedirectAllowed` has been added to the HTTP section of the JIS server INI file:

```
[HTTP]
RelativeRedirectAllowed=1
```

The `RelativeRedirectAllowed` property is enabled by default.

New MakeJar Server Configuration Properties

In order to allow more flexibility in the code signing process, the following options have been added to the `MakeJar` section of the server INI file:

Table 4 - 12 : JIS 9.2.4 New MakeJar Server Configuration Propeties

Property Name	Descrption
certchain	Name of alternative certchain file `certchain=<file>`

Table 4 - 12 : JIS 9.2.4 New MakeJar Server Configuration Propeties

Property Name	Descrption
sigfile	Name of .SF/.DSA file `sigfile=<file>`
signedjar	Name of signed JAR file `signedjar=<file>`
digestalg	Name of digest algorithm `digestalg=<algorithm>`
sigalg	Pame of signature algorithm `sigalg=<algorithm>`
tsacert	Public key certificate for Timestamping Authority `tsacert=<alias>`
tsapolicyid	TSAPolicyID for Timestamping Authority `tsapolicyid=<oid>`
tsadigestalg	Algorithm of digest data in timestamping request `tsadigestalg=<algorithm>`
altsigner	Class name of an alternative signing mechanism `altsigner=<class>`
altsignerpath	Location of an alternative signing mechanism `altsignerpath=<pathlist>`
sectionsonly	Don't compute hash of entire manifest `sectionsonly=0`

Table 4 - 12 : JIS 9.2.4 New MakeJar Server Configuration Propeties

Property Name	Descrption
internalsf	Include the .SF file inside the signature block `internalsf=0`
protected	Keystore has protected authentication path `protected=0`
providerName	Provider name `providerName=<name>`
providerClass	Name of cryptographic service provider's master class file and constructor argument `providerClass=<class>`
providerArg	Optional providerClass constructor argument `providerArg=<arg>`
strict	Treat warnings as errors `strict=0`
revCheck	Enable certificate revocation check `revCheck=0`
addprovider	Add security provider by name (e.g. SunPKCS11) `addprovider=<name>`
providerArg	Optional security providerconfigure argument `providerArg<arg>`
javaOption	Specify a JVM option or argument `javaOption=<arg>`

Table 4 - 12 : JIS 9.2.4 New MakeJar Server Configuration Propeties

Property Name	Descrption
conf	Specify a pre-configured options file `conf=<url>`
addOptions	Specify an additional jarsigner option or argument `addOptions=<arg>`

With the exception of the `javaOption` and `addOptions` properties, these properties directly correspond to jarsigner command line options. For example, the `certchain` property corresponds to the `-certchain` option. For those options that take arguments, the value of the property corresponds to the argument for the option. Using the `-certchain` option as an example, the server INI file setting:

```
[MakeJar]certchain=C:/CERTS/chain.crt
```

would result in the following arguments being appended to the jarsigner comand line:

```
-certchain c:\CERTS\chain.crt
```

The `internalsf`, `sectionsonly`, `protected`, `strict` and `revCheck` properties correspond to boolean command line options with values of 1 or 0 indicating the option is enabled or disabled. For example,

```
[MakeJar]strict=1
```

would enable the `-strict` command line option. All of these properties are disabled by default.

The command line options corresponding to the `revCheck`, `addProvider` and `conf` properties are new for Java 17 and not supported on Java 8.

The `javaOption` property corresponds to the `-J<javaOption>` Java VM command line option and `<javaOption>` is passed through to the Java VM running the jarsigner command.

The `addOptions` property does not correspond to any jarsigner command line options. If set, it is appended to the jarsigner command line.

New Application Configuration Properties

Support for the following application configuration properties have been added in JIS 9.2.4:

- EnableTCPKeepAlive
- SupportedProtocols
- SupportedCipherSuites

EnableTCPKeepAlive

The `EnableTCPKeepAlive` property in the `GUISys TN3270` and `GUISys TN5250` sections allow the user to enable the operating system `SO_KEEPALIVE` feature for server/host socket connections.

> **Note:** The `SO_KEEPALIVE` feature is platform dependent and KeepAlive timeouts must be set at the O/S level.

SupportedProtocols and SupportedCipherSuites

The `SupportedProtocols` and `SupportedCipherSuites` properties in the `GUISys TN3270` and `GUISys TN5250` sections of the application INI file can be used to configure the SSL protocols and cipher suites enabled for SSL communication between JIS server and the host.

The value of the `SupportedProtocols` property will be used to set the protocols enabled for communication between the JIS server and the host. If the `SupportedProtocols` property is unset, the default protocols supported by the JVM, excluding SSLv3 and SSLv2Hello will be enabled.

The value of the `SupportedCipherSuites` property will be used to set the cipher suites enabled for communication between the JIS server and the host. If the `SupportedCipherSuites` property is unset, the default cipher suites supported by the JVM will be used.

The default value for the `SupportedProtocols` and `SupportedCipherSuites` properties is unset.

Note: The `SupportedProtocols` and `SupportedCipherSuites` properties in the `GUISys TN3270` and `GUISys TN5250` sections only apply to server / host communication and do not replace or affect the settings of the `SupportedProtocols` or `SupportedCipherSuites` properties client / server commnuication in the `GeneralParameters` or `HTTP` sections of the server INI file.

TLSv1.3 requires Java 8u261 or later.

Example5. JIS 9.2.4 GUiSys TN3270 Property Examples

```
[GUISys TN3270]
;SecureHostConnection=1
;SupportedProtocols=TLSv1.2,TLSv1.3
;SupportedCipherSuites=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_DHE_DSS
_WITH_AES_128_GCM_SHA256
;EnableTCPKeepAlive=0
```

Example6. JIS 9.2.4 GUiSys TN5250 Property Examples

```
[GUISys TN5250]
;SecureHostConnection=1
;SupportedProtocols=TLSv1.2,TLSv1.3
;SupportedCipherSuites=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_DHE_DSS
_WITH_AES_128_GCM_SHA256
;EnableTCPKeepAlive=0
```

Detailed Description of Fixes Included in JIS 9.2.4

The following fixes are included in JIS 9.2.4:

- JIS-2029 / PI-1292251 - 9.2.2 - Compile Error after upgrading
  JIS-2224 - JNLP Servlet log should be created in the directory specified by RtLogDir or -l startup option
- JIS-2157 / PI-1356981 - JIS Administrator not showing ini settings
  JIS-2162 - The jrodefaults.ini file is not installed in the runtime installation
- JIS-2220 / PI-5391880 - 404 error when migrating to new upgraded prod
  JIS-2221 - Add support for server / host SSL protocol and cipher suite settings
- JIS-2318 / PI-5441448 - Paste with a emdash causes disconnect
  JIS-2340 - Remove em-dash and en-dash unicode characters from data sent to host

- JIS-2338 / PI-5444847 - Multi-browser Support XhtmlSubapplGennerator:error Null Pointer
  JIS-2343 - Add checks for null pointers and improve logging.
- JIS-2346 / PI-5452198 - Action box not reflecting String resources localisation files
  JIS-2374 - Include radio button text when doing translation for localization
- JIS-2352 / PI-1458958 - Application Menu
  JIS-2522 - Handle NullPointerException when running a JWS Java Client Application with RunInsideBrowser=TRUE
- JIS-2357 / PI-5457849 - Blackduck Code scan
  JIS-2361 - CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page
  JIS-2362 - Validate DebugLevel in AbstractJAMServlet
  JIS-2363 - Remove dependence on URL query string for JIS session information
  JIS-2366 - Update embedded Jetty to version 9.4.44
- JIS-2368 / SI-450620 - Null pointer exception when using JNLP
  JIS-2371 - Revert access change for RunTimeApplication InitBeforeFirstSubAppl methods
- JIS-2375 / SI-452878 - Hotfix 3216 and Session closed message
  JIS-2380 - KeepAlive requests are failing on chrome when language is unset
- JIS-2377 / SI-452781 - Content-Type charset setting
  JIS-2424 - Add JIS server property setting to allow configuring Content-Type header charset
- JIS-2382 / SI-453775 - Need impact on the product logging due to Log4J vulnerability|Create a patch for 9.2.2 and 9.2.3 that will exclude the JMSAppender class.
  JIS-2383 - Remove JMSAppender class from Natural Parser log4j.jar
- JIS-2384 / SI-454668 - Web-Cache Poisoning
  JIS-2474 - Add RelativeRedirectAllowed to Jetty config
- JIS-2395 / SI-461620 - Client is Reporting Latency Since Deploying Multi-Browser Changes
  JIS-2402 - Remove extraneous debug logging from the com.jacada.jis.runtime.server.http.ActionHandler class
- JIS-2396 / SI-462140 - Session timeout message after 60 seconds sitting idle
  JIS-2401 - Fix parsing of URL query string for JBS parameter
- JIS-2397 / SI-460216 - Jacada Administrator - Connection Closed error.
  JIS-2398 - Fix ClassCastException thrown from JAM while fetching INI file params.
  JIS-2399 - jrodefaults.ini is not included in the JIS runtime installation
  JIS-2400 - Improve error handling for missing jrodefaults.ini file
- JIS-2408 / SI-463941 - Source scan Issues in jam.jar and jacada Files
  JIS-2489 / SI-503160 - Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG) (CWE ID 338)
  JIS-2410 - Allow JRE to specify SecureRandom algorithm

JIS-2446 - Investigate and if possible remove instances of deserialization of untrusted data

- JIS-2416 / SI-468651 - Uncaught type error in Jacada.js and prototype.js
  JIS-2425 - Fix uncaught TypeError in prototype.js
- JIS-2417 / SI-468446 - Jacada 9.2.3 - Jacada Claim
  JIS-2420 - Improve debug logging
  JIS-2421 - Global variables for screen interpreter only initialized for first session
- JIS-2427 / SI-474156 - Error with admin.jnlp
  JIS-2433 - Add support for running JAM as a JNLP application
- JIS-2436 / SI-476930 - Jacada Production Issue
  JIS-2439 - Fix JMX Command Line Operations suspend method
- JIS-2437 / SI-477423 - Increase trace level in JBS file
  JIS-2441 - Enable setting of SessionCoreDump properties via server INI in addition to appl ini.
- JIS-2440 / SI-477310 - Need ACE.exe (JIS jdk) signed
  JIS-2445 - The InstallShield post build batch file is not being run during JIS trunk builds
- JIS-2451 / SI-485344 - ACE.exe digital cert expired
  JIS-2452 - Update JIS code signing certificate for 2022-2023
- JIS-2456 / SI-489457 - Java update and JIS
  JIS-2458 - Update JIS Java build version
- JIS-2460 / SI-491960 - Jetty Log is growing very large on server
  JIS-2461 - Add support for configuring jetty logging
- JIS-2471 / SI-498702 - CTE specs for BBI
  JIS-2472 - Add jetty version to JIS server log
- JIS-2475 / SI-501334 - Delta Air Lines - JIS Vulnerability Scan
  JIS-2476 / SI-503192 - J2EE Bad Practices: Use of System.exit() (CWE ID 382)
  JIS-2515 - Mitigate use of System.exit() (CWE ID 382)
- JIS-2477 / SI-503193 - Improper Resource Shutdown or Release (CWE ID 404)
  JIS-2497 - Fixes for improper resource shutdown (CWE ID-404)
- JIS-2479 / SI-503157 - Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) (CWE ID 80)
  JIS-2519 - Sanitize URL used for generating JAM JNLP file
- JIS-2485 / SI-503161 - Use of a Broken or Risky Cryptographic Algorithm (CWE ID 327)
  JIS-2531 / SI-520395 - Production Issue - Users receiving blank screen
  JIS-2502 - Add logging for debugging redirection proxy
  JIS-2516 - Replace DES with AES algorithm
- JIS-2486 / SI-503162 - External Control of File Name or Path (CWE ID 73)
  JIS-2512 - Fix filename for vulnerability
- JIS-2487 / SI-503163 - Deserialization of Untrusted Data (CWE ID 502)
  JIS-2544 - Resolve deserialization issue (CWE Id-502)

- JIS-2491 / SI-503168 - Improper Restriction of XML External Entity Reference (CWE ID 611)
  JIS-2514 - Disable external entity reference in the XML document builder
- JIS-2494 / SI-503172 - Selection of Less-Secure Algorithm During Negotiation ('Algorithm Downgrade') (CWE ID 757)
  JIS-2511 - Disable TLS1.1
- JIS-2495 / SI-503507 - Library XHTML Template Extensions Aren't Merged
  JIS-2496 - Fix merging of library user extensions
- JIS-2504 / SI-505992 - Security Vulnerabilities for TLS Protocol and Cipher Suites
  JIS-2520 - Apply SupportedProtocols and SupportedCipherSuites properties to second JIS SSL server port
- JIS-2517 / SI-510751 - Infinite Loop problem SI449095 came back with latest hotfix
  JIS-2518 - Fix infinite loop problem for invalid dates
- JIS-2527 / SI-518330 - Issues that reported in Blackduck scan to be fixed
  JIS-2528 - Upgrade embedded Jetty to version 9.4.52
- JIS-2534 / SI-523094 - JIS Secure Telnet - TLS Protocol
  JIS-2535 - Add logging of negotiated SSL protocol and cipher suite used for SSL connections
- JIS-2536 / SI-523570 - New vulnerability found with Jetty
  JIS-2537 - Upgrade embedded Jetty to version 9.4.53
- JIS-2548 / SI-551097 - Penetration Scan has Identified Predictable Session Tokens Vulnerability
  JIS-2569 - Upgrade Session IDs to 128 bits using UUID
- JIS-2550 / SI-552350 - Issue with Keep Alive
  JIS-2562 - Add application property setting to OS level TCPKeepAlive
- JIS-2580 / SI-564462 - Need help getting admin tool working on the Linux servers.
  JIS-2595 - Modify the method for Deserialization
- JIS-2581 / SI-564791 - Need Explanation of how ports work
  JIS-2584 - Improve logging for Xhtml Client Redirection Proxy
  JIS-2587 - Adding support for enabling the SupportHTTPSOnly setting in HTTP section for Redirection Proxy
- JIS-2588 / SI-567394 - Jacada Cert Issue
  JIS-2590 - Add properties for additional jarsigner command line options
- JIS-2602 / SI-569157 - JIS 9.2.4 beta release attachment
  JIS-2608 - Support for Java command line utils on Java 17
  JIS-2609 - Upgrade IzPack for Java 17 support
  JIS-2610 - Update batch files for unbundled JVM or JDK

ACE

JIS-2157 / PI-1356981 - JIS Administrator not showing ini settings
JIS-2162 - The jrodefaults.ini file is not installed in the runtime installation
JIS-2397 / SI-460216 - Jacada Administrator - Connection Closed error.
JIS-2399 - jrodefaults.ini is not included in the JIS runtime installation

Display of the JAM ini settings requires the jrodefaults.ini file be present in the runtime installation. It is now included in the runtime installation.

JIS-2382 / SI-453775 - Need impact on the product logging due to Log4J vulnerability|Create a patch for 9.2.2 and 9.2.3 that will exclude the JMSAppender class.
JIS-2383 - Remove JMSAppender class from Natural Parser log4j.jar

A vulnerability (CVE-2021-4104, deserialization of untrusted data) was found in the version of Log4J shipped with the Natual Parser included in JIS. Removal of the JMSAppender class from the log4j.jar file mitigates this vulnerability.

JIS-2451 / SI-485344 - ACE.exe digital cert expired
JIS-2452 - Update JIS code signing certificate for 2022-2023

The certificate used to sign the JIS executables and jar files expired, requiring them to be re-signed with a new certificate.

JIS-2588 / SI-567394 - Jacada Cert Issue
JIS-2590 - Add properties for additional jarsigner command line options

New properties were added to the JIS server INI file `[MakeJar]` section to to support generation of signed application jar files with newer keystore providers.

JIS-2602 / SI-569157 - JIS 9.2.4 beta release attachment
JIS-2608 - Support for Java command line utils on Java 17
JIS-2609 - Upgrade IzPack for Java 17 support
JIS-2610 - Update batch files for unbundled JVM or JDK

JIS Java code compilation is now supported on Java 8 or Java 17.

The IzPack runtime installation generation utility has been upgraded to version 5.2.3 to allow generation of JIS runtime installations using Java 8 or Java 17.

The JIS batch utilities have been updated to make them easier to modify for use with an unbundled Java JDK or JVM.

JAM

JIS-2357 / PI-5457849 - Blackduck Code scan
JIS-2362 - Validate DebugLevel in AbstractJAMServlet

The debug level parameter derived from the URL query string used to start JAM is now validated as an integer between 0 and 200. Invalid debug level settings are logged and the debug level is set to the default value of 1.

JIS-2397 / SI-460216 - Jacada Administrator - Connection Closed error.
JIS-2398 - Fix ClassCastException thrown from JAM while fetching INI file params.
JIS-2400 - Improve error handling for missing jrodefaults.ini file

The error message for a missing jrodefaults.ini file has been improved and now indicates the file was not found.

> **Note:** JAM logging is disabled by default and must be enabled to view the log.

Error handling for a missing jrodefaults.ini file during JAM startup, has been improved.

JIS-2408 / SI-463941 - Source scan Issues in jam.jar and jacada Files
JIS-2446 - Investigate and if possible remove instances of deserialization of untrusted data

JAM related deserialization vulnerabilities have been resolved.

JIS-2427 / SI-474156 - Error with admin.jnlp
JIS-2433 - Add support for running JAM as a JNLP application

The JNLP file used to start JAM has been updated allowing JAM to be started via JNLP on Open Web Start as well as Java Web Start. Prior to this fix, JAM could only run via JNLP as an applet which is not supported on Open Web Start.

JIS-2451 / SI-485344 - ACE.exe digital cert expired
JIS-2452 - Update JIS code signing certificate for 2022-2023

The certificate used to sign the JIS executables and jar files expired, requiring them to be re-signed with a new certificate.

JIS-2456 / SI-489457 - Java update and JIS
JIS-2458 - Update JIS Java build version

Oracle Java 8u351 introduced a change in the way that signed jar files are treated for Java Web Start / JNLP applications. Starting with Java 8u351, Jar files signed with SHA-1 algorithms are now treated as unsigned, preventing them from being loaded via JNLP. This, in turn prevents them from being used in JIS Java Client applications started via Java Web Start / JNLP. The JIS clbase and jam jars are now signed with SHA-256 algorithms, which are compatible with Java 8u351.

Note: Client application jar files must be signed with the jarsigner command from Java 8u101 or newer versions of java. The Java version (Java 8u102) shipped with JIS 9.2.3 signs jar files with SHA-256 algorithms by default and is compatible with Java 8u351.

JIS-2580 / SI-564462 - Need help getting admin tool working on the Linux servers.
JIS-2595 - Modify the method for Deserialization

A JAM/server deserialization problem which prevented starting JAM from a remote server has been resolved.

Java client

JIS-2352 / PI-1458958 - Application Menu
JIS-2522 - Handle NullPointerException when running a JWS Java Client Application with RunInsideBrowser=TRUE

A problem which caused a `NullPointerException` to be thrown when starting a JIS Java Client application via JavaWebStart with `RunInsideBrowser=TRUE` has been resolved

JIS-2408 / SI-463941 - Source scan Issues in jam.jar and jacada Files
JIS-2446 - Investigate and if possible remove instances of deserialization of untrusted data

Java Client related deserialization vulnerabilities have been resolved.

JIS-2451 / SI-485344 - ACE.exe digital cert expired
JIS-2452 - Update JIS code signing certificate for 2022-2023

The certificate used to sign the JIS executables and jar files expired, requiring them to be re-signed with a new certificate.

JIS-2456 / SI-489457 - Java update and JIS
JIS-2458 - Update JIS Java build version

Oracle Java 8u351 introduced a change in the way that signed jar files are treated for Java Web Start / JNLP applications. Starting with Java 8u351, Jar files signed with SHA-1 algorithms are now treated as unsigned, preventing them from being

loaded via JNLP. This, in turn prevents them from being used in JIS Java Client applications started via Java Web Start / JNLP. The JIS clbase and jam jars are now signed with SHA-256 algorithms, which are compatible with Java 8u351.

> **Note:** Client application jar files must be signed with the jarsigner command from Java 8u101 or newer versions of java. The Java version (Java 8u102) shipped with JIS 9.2.3 signs jar files with SHA-256 algorithms by default and is compatible with Java 8u351.

JIS-2479 / SI-503157 - Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) (CWE ID 80)
JIS-2519 - Sanitize URL used for generating JAM JNLP file

The URL used to generate the JAM JNLP file is sanitized with the OWASP HTML santizer utility prior to use.

JIS-2548 / SI-551097 - Penetration Scan has Identified Predictable Session Tokens Vulnerability
JIS-2569 - Upgrade Session IDs to 128 bits using UUID

Client Session IDs are now generated as 128-bit UUID values.

JIS-2602 / SI-569157 - JIS 9.2.4 beta release attachment
JIS-2608 - Support for Java command line utils on Java 17

JIS Java code compilation is now supported on Java 8 or Java 17.

Server

JIS-2029 / PI-1292251 - 9.2.2 - Compile Error after upgrading
JIS-2224 - JNLP Servlet log should be created in the directory specified by RtLogDir or -l startup option

The JIS JNLP Servlet log is now created in the directory specified by the -l option to the JIS server startup script or by the `[General]RtLogsDir` property setting in the JIS server INI file.

JIS-2220 / PI-5391880 - 404 error when migrating to new upgraded prod
JIS-2221 - Add support for server / host SSL protocol and cipher suite settings

Two new application config options have been added in the GUISys TN320 and GUISys Tn5250 sections of the application INI file: SupportedProtocols and SupportedCipherSuites. These can be used to configure SSL communication between the JIS server and host.

JIS-2318 / PI-5441448 - Paste with a emdash causes disconnect
JIS-2340 - Remove em-dash and en-dash unicode characters from data sent to host

Em-dash and en-dash Unicode characters are now stripped from data sent from the server to the host to prevent host disconnect.

JIS-2338 / PI-5444847 - Multi-browser Support XhtmlSubapplGennerator:error Null Pointer
JIS-2343 - Add checks for null pointers and improve logging.

Add checks to prevent NullPointerExceptions during XHTML response generation.

JIS-2357 / PI-5457849 - Blackduck Code scan
JIS-2361 - CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page
JIS-2363 - Remove dependence on URL query string for JIS session information
JIS-2366 - Update embedded Jetty to version 9.4.44

The embedded Jetty was upgraded to version 9.4.44.

JIS-2368 / SI-450620 - Null pointer exception when using JNLP
JIS-2371 - Revert access change for RunTimeApplication InitBeforeFirstSubAppl methods

A change that prevented the UserInitBeforeSubAppl general extention method from being run was reverted.

JIS-2375 / SI-452878 - Hotfix 3216 and Session closed message
JIS-2380 - KeepAlive requests are failing on chrome when language is unset

Aproblem with Xhtml client KeepAlive request on Chrome browsers introduced with build JIS 9.2.3 hotfix build 3216, was fixed.

JIS-2377 / SI-452781 - Content-Type charset setting
JIS-2424 - Add JIS server property setting to allow configuring Content-Type header charset

A server config property was added to allow setting the charset HTTP Content-Type header directive for static HTML pages.

JIS-2384 / SI-454668 - Web-Cache Poisoning
JIS-2474 - Add RelativeRedirectAllowed to Jetty config

A configuration change has been made to the JIS Jetty HTTP server. The RelativeRedirectionAllowed setting is now enabled by default. When enabled, RelativeRedirectAllowed allows the HTTP server to redirect URLs for server directories which do not end in a slash to be redirected to a relative location.

JIS-2395 / SI-461620 - Client is Reporting Latency Since Deploying Multi-Browser Changes
JIS-2402 - Remove extraneous debug logging from the com.jacada.jis.runtime.server.http.ActionHandler class

Extraneous debug logging was remvoed from the ActionHandler class.

JIS-2396 / SI-462140 - Session timeout message after 60 seconds sitting idle
JIS-2401 - Fix parsing of URL query string for JBS parameter

A problem with communication between the Xhtml client and the JIS Redirection Proxy has been resolved.

JIS-2408 / SI-463941 - Source scan Issues in jam.jar and jacada Files
JIS-2489 / SI-503160 - Use of Cryptographically Weak Pseudo-

Random Number Generator (PRNG) (CWE ID 338)
JIS-2410 - Allow JRE to specify SecureRandom algorithm
JIS-2446 - Investigate and if possible remove instances of deserialization of untrusted data

Configuration of the random number generator was modified to allow the system to chose the "best" available PNRG instead of selecting a specific algorithm.

JAM and Java Client related deserialization vulnerabilities have been resolved.

JIS-2417 / SI-468446 - Jacada 9.2.3 - Jacada Claim
JIS-2420 - Improve debug logging
JIS-2421 - Global variables for screen interpreter only initialized for first session

Debug logging improvements for INI file processing has been improved.

A regresssion in global variable initilization has been fixed.

JIS-2427 / SI-474156 - Error with admin.jnlp
JIS-2433 - Add support for running JAM as a JNLP application

The JNLP file used to start JAM has been updated to make use of the <application-descr> tag instead of the <applet-descr> to allow running JAM via OpenWebStart.

JIS-2436 / SI-476930 - Jacada Production Issue
JIS-2439 - Fix JMX Command Line Operations suspend method

A typo in the JAM JMX suspend command has been fixed.

It is now possible to suspend the JIS server via JAM JMX commands.

JIS-2437 / SI-477423 - Increase trace level in JBS file
JIS-2441 - Enable setting of SessionCoreDump properties via server INI in addition to appl ini.

Enable setting of the SessionCoreDump properties: AlwaysDump, ClientCommunicationFailureDump, DumpFast and DumpDumpOnInactivityTimeout properities in the server INI file in addition to being able to set them in the application INI file.

> **Note:** SessionCoreDump property settings in the application INI file override settings in the server INI file.

JIS-2451 / SI-485344 - ACE.exe digital cert expired
JIS-2452 - Update JIS code signing certificate for 2022-2023

The certificate used to sign the JIS executables and jar files expired, requiring them to be re-signed with a new certificate.

JIS-2460 / SI-491960 - Jetty Log is growing very large on server
JIS-2461 - Add support for configuring jetty logging

The Jetty log level is now configurable via a properties file. Jetty logging is set to WARN by default but can be configured by modifying the jetty-logging.properties file included in the JacadaFiles\classes directory. For runtime installations, copy the jetty-logging.properties file to the classes directory in the runtime installation. The Jetty log file is overwritten each time the JIS server is started.

The jetty-logging.properties file is now included in theruntime installation

JIS-2471 / SI-498702 - CTE specs for BBI
JIS-2472 - Add jetty version to JIS server log

The Jetty version is now included in the JIS server logs.

JIS-2475 / SI-501334 - Delta Air Lines - JIS Vulnerability Scan
JIS-2476 / SI-503192 - J2EE Bad Practices: Use of System.exit() (CWE ID 382)
JIS-2515 - Mitigate use of System.exit() (CWE ID 382)

Usage of the System.exit() method, which could shut down the web application server have been replaced in the J2EE web application version of JAM. An exception is thrown to the container instead of exiting.

JIS-2477 / SI-503193 - Improper Resource Shutdown or Release (CWE ID 404)
JIS-2497 - Fixes for improper resource shutdown (CWE ID-404)

Instances of improper resource shutdown or release have been fixed.

JIS-2479 / SI-503157 - Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) (CWE ID 80)
JIS-2519 - Sanitize URL used for generating JAM JNLP file

The URL used to generate the JAM JNLP file is sanitized with the OWASP HTML santizer utility prior to use.

JIS-2485 / SI-503161 - Use of a Broken or Risky Cryptographic Algorithm (CWE ID 327)
JIS-2531 / SI-520395 - Production Issue - Users receiving blank screen
JIS-2502 - Add logging for debugging redirection proxy
JIS-2516 - Replace DES with AES algorithm

The DES algorithm used for encrypting URLs in JIS XHTML client server communication has been replaced with the AES-128 algorithm.

JIS-2486 / SI-503162 - External Control of File Name or Path (CWE ID 73)
JIS-2512 - Fix filename for vulnerability

The HTML file name used for print jobs is now sanitized to mitigate a possible security vulnerability.

JIS-2487 / SI-503163 - Deserialization of Untrusted Data (CWE ID 502)
JIS-2544 - Resolve deserialization issue (CWE Id-502)

A JAM/server deserialization vulnerability has been resolved.

JIS-2491 / SI-503168 - Improper Restriction of XML External Entity Reference (CWE ID 611)
JIS-2514 - Disable external entity reference in the XML document builder

External entity references are no longer supported by the JIS server XML parsers.

JIS-2494 / SI-503172 - Selection of Less-Secure Algorithm During Negotiation ('Algorithm Downgrade') (CWE ID 757)
JIS-2511 - Disable TLS1.1

SSL/TLSv1.1 and older protocols are no longer supported by the JIS Server. Java clients and legacy hosts must support TLSv1.2 or higher.

JIS-2495 / SI-503507 - Library XHTML Template Extensions Aren't Merged
JIS-2496 - Fix merging of library user extensions

A bug introduced with the fix for JIS-2375 which prevented merging of user library extensions with generated HTML has been fixed.

JIS-2504 / SI-505992 - Security Vulnerabilities for TLS Protocol and Cipher Suites
JIS-2520 - Apply SupportedProtocols and SupportedCipherSuites properties to second JIS SSL server port

See JIS-2494 for details.

JIS-2527 / SI-518330 - Issues that reported in Blackduck scan to be fixed
JIS-2528 - Upgrade embedded Jetty to version 9.4.52

The embedded Jetty was upgraded to version 9.4.52 to resolve a security vulnerability.

JIS-2534 / SI-523094 - JIS Secure Telnet - TLS Protocol
JIS-2535 - Add logging of negotiated SSL protocol and cipher suite used for SSL connections

The SSL protocol and cipher suite negotiated for client/server and server/host connections are now logged.

JIS-2536 / SI-523570 - New vulnerability found with Jetty
JIS-2537 - Upgrade embedded Jetty to version 9.4.53

Upgraded embedded Jetty to version 9.4.53 to address CVE-2023-36478 and CVE-2023-44487.

JIS-2548 / SI-551097 - Penetration Scan has Identified Predictable Session Tokens Vulnerability
JIS-2569 - Upgrade Session IDs to 128 bits using UUID

Client Session IDs are now generated as 128-bit UUID values.

JIS-2550 / SI-552350 - Issue with Keep Alive
JIS-2562 - Add application property setting to OS level TCPKeepAlive

A new application INI property setting has been added to allow enabling O/S TCP Keepalive.

JIS-2580 / SI-564462 - Need help getting admin tool working on the Linux servers.
JIS-2595 - Modify the method for Deserialization

A JAM/server deserialization problem which prevented starting JAM froma remote server has been resolved.

JIS-2581 / SI-564791 - Need Explanation of how ports work
JIS-2584 - Improve logging for Xhtml Client Redirection Proxy
JIS-2587 - Adding support for enabling the SupportHTTPSOnly setting in HTTP section for Redirection Proxy

Logging for the Xhtml Client Redirection Proxy has been improved to aid in debugging Xhtml Client / Server communication problems.

Support for HTTPS communiction between the Xhtml Client Redirection Proxy and JIS server process has been added.

JIS-2602 / SI-569157 - JIS 9.2.4 beta release attachment
JIS-2610 - Update batch files for unbundled JVM or JDK

The JIS batch utilities have been updated to make them easier to modify for use with an unbundled Java JDK or JVM.

XHTML Client

JIS-2346 / PI-5452198 - Action box not reflecting String resources localisation files
JIS-2374 - Include radio button text when doing translation for localization

A problem with localized string resources for Action box radio buttons has been resolved.

JIS-2357 / PI-5457849 - Blackduck Code scan
JIS-2361 - CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page
JIS-2363 - Remove dependence on URL query string for JIS session information

JIS XHTML Client applications no longer make use of the HTML query string to pass the JIS Session ID between the client and server.

JIS-2375 / SI-452878 - Hotfix 3216 and Session closed message
JIS-2380 - KeepAlive requests are failing on chrome when language is unset

The XHTML Client KeepAlive mechanismwas modifed to check for a missing language setting and log an error but continue processing.

JIS-2377 / SI-452781 - Content-Type charset setting
JIS-2424 - Add JIS server property setting to allow configuring Content-Type header charset

A new JIS server INI file property: `[HTTP]ContentTypeCharset`, was added to allow configuring the HTTP Content-Type header.

JIS-2396 / SI-462140 - Session timeout message after 60 seconds sitting idle
JIS-2401 - Fix parsing of URL query string for JBS parameter

A problem with communication between the Xhtml client and the JIS Redirection Proxy has been resolved.

JIS-2416 / SI-468651 - Uncaught type error in Jacada.js and prototype.js
JIS-2425 - Fix uncaught TypeError in prototype.js

A problem which caused an uncaught TypeError to be thrown from prototype.js has been resolved.

JIS-2517 / SI-510751 - Infinite Loop problem SI449095 came back with latest hotfix
JIS-2518 - Fix infinite loop problem for invalid dates

A problem where entering a date with a day greater than the number of days in the month, where the month has less than 31 days (February, April, June, September or November) would cause the date control validation to go into an infinite loop has been resolved.

JIS-2548 / SI-551097 - Penetration Scan has Identified Predictable Session Tokens Vulnerability
JIS-2569 - Upgrade Session IDs to 128 bits using UUID

Client Session IDs are now generated as 128-bit UUID values.

JIS-2581 / SI-564791 - Need Explanation of how ports work
JIS-2584 - Improve logging for Xhtml Client Redirection Proxy
JIS-2587 - Adding support for enabling the SupportHTTPSOnly setting in HTTP section for Redirection Proxy

Logging for the Xhtml Client Redirection Proxy has been improved to aid in debugging Xhtml Client / Server communication problems.

Support for HTTPS communiction between the Xhtml Client Redirection Proxy and JIS server process has been added.

Limitations of JIS 9.2.4

The following are known limitations in JIS 9.2.4:

- JIS-2413 / SI-466468 - ContentSecurityPolicy vulnerability

JIS-2413 / SI-466468 - ContentSecurityPolicy vulnerability

If ContentSecurityPolicy settings are enabled, policy settings which do not include:

```
script-src 'unsafe-inline' 'unsafe-eval'; style-src 'unsafe-inline
```

will likely cause problems for JIS XHTML client applications.

JIS Interface Server 9.2.3 Release Notes

Installation & Upgrade Information

Supported Platforms

JIS Interface Server 9.2.3 is supported on the following platforms:

- Windows Server 2016 Standard and Enterprise (64-bit)
- Windows Server 2019 Standard and Enterprise (64-bit)
- Windows 10 Professional, Ultimate and Enterprise Edition (64-bit)
- Solaris SPARC 11 (64-bit)
- AIX 7.1 Power (64-bit)
- AIX 7.2 Power (64-bit)
- Red Hat Enterprise Linux 7 for x86 (64-bit)
- Red Hat Enterprise Linux 8 for x86 (64-bit)

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for Java versions, browser versions and application server versions supported by their respective vendors. Being so, when a vendor no longer supports a Java version, browser version or application server version, Software GmbH will discontinue supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows 10
- Windows Server 2019

Clients

Java Client Applets

The Java Client Applet has been tested on the following operating systems, browser and Java versions:

- Windows 10

Table 5 - 13 : JIS 9.2.3 Java Applet Client Supported Browser and Java Versions

Browser	JRE
IE 11	Oracle 1.8.0

Support for the Java Plugin (required to launch Java applets in a browser) has been dropped from Chrome, Firefox and Microsoft Edge, effectively removing them from the list of browsers supported for Java client applets. With support for Internet Explorer being dropped June 15, 2022, Java client applet users are strongly urged to migrate to using JNLP to launch Java clients. See “Open Web Start Support” on page 75 for more details. Software GmbH will end support of Java client applets with the end of support for Internet Explorer.

Java Web Start Clients

The Java Web Start Client has been tested on the following operating systems, browser and Java versions:

- Windows 10
- Windows Server 2018

Table 5 - 14 : JIS 9.2.3 Java Web Start Client Supported Browser and Java Versions

Browser	JRE
Chrome	Oracle 1.8.0, Azul Zulu Java 8
Edge	Oracle 1.8.0, Azul Zulu Java 8
Firefox	Oracle 1.8.0, Azul Zulu Java 8
IE	Oracle 1.8.0, Azul Zulu Java 8

XHTML Clients

The XHTML client has been tested with the following operating system and browser versions:

Table 5 - 15 : JIS 9.2.3 XHTML Client Supported Operating Systems and Browsers

Operating Syste	Browser
Windows 10	Chrome, Edge, Firefox, IE 11
Windows Server 2019	Chrome, Edge, IE 11
Red Hat Linux 8	Firefox
iPadOs 14.5.1	Chrome, Safari

Standalone Server

The JIS standalone server has been tested in the following environments:

Table 5 - 16 : JIS 9.2.3 Standalone Server Supported Environments

Operating System	Java Version
Windows 10	Oracle 1.8.0, Azul Zulu Java 8
Windows Server 2019	Oracle 1.8.0, Azul Zulu Java 8
Red Hat Linux AS8	Oracle 1.8.0, Azul Zulu Java 8

Installation and Upgrade Instructions

The following steps describe the procedure for installing JIS Interface Server 9.2.3 and upgrading from a previous version.

Note: When installing JIS 9.2.3 on Windows, the JIS installer must be run with administrator permissions.

Installing JIS Interface Server 9.2.3

- Copy the CD key from the RTCP section of the `ace.ini` or `ace400.ini` file in root directory of the previous JIS installation: `<JIS>\ace.ini` for mainframe installations or `<JIS>\ace400.ini` for AS/400 installations:

```
[RTCP]
CDKey=<JIS_CDKey>
```

- Install JIS 9.2.3 in a new directory. Do not install over the top of a previous version of JIS.
- Install Azul Open JRE/JDK Java 8

Pack JIS application from previous version of JIS Interface Server

- Open the application in the ACE Designer in the previous version of JIS
- Select Utility/Pack from the menu
- Follow the prompts in the pack wizard:
  - Select all libraries and subapplications

    - Select Single File
    - Select Java Files
    - Select all checkboxes for DDS, SDF, and Screens
    - Uncheck Temporary Auxiliary Files and Installations
    - Select Don't pack any configuration files
    - Include any extra image files that are not found in the <Application>images folder
    - Include any additional files being used by the application that are not default to JIS. Examples are 3rd party software, jar files, etc.
- Allow the Pack Process to complete

Unpack the Jpack in JIS Interface Server 9.2.3

- Open the JIS ACE Designer
- Select Utility/Unpack
- Follow the prompts in the Unpack Wizard
    - Select the jpack that was created from the previous version of JIS by clicking on the Browse button
    - Select everything under items to unpack. By default everything will be selected
    - Select Packed Application
    - Compare Library Names to make sure the original and new library names match
    - Select java files
    - Select DDS, SDF, Screens from Input Directories
    - Select Runtime Files
    - Select Don't Unpack Any Configuration files
    - Select … to point any 3rd party files and images into the new location
- Click finish

Recompile the JIS Interface Server Application

- Open the JIS ACE Designer
- Select File/Open Application and open the application
- Select File/Generate Runtime and follow the prompts in the wizard
    - Select Runtime Type Java, XHTML or both
    - Select Server Platform Windows, Linux, Solaris, AIX
    - Select Libraries to include
    - Select all libraries to recompile
    - Select all subapplications to include in runtime

  - Select all subapplications to process
  - Add your host name or IP address and port number
- Click Finish

Recompile any Java extensions

- Open the Windows File Explorer and select `<JIS9.2.3>\jacadafiles\src\appls\<Application>`
- In the `<JIS9.2.3>\jacadafiles\src\appls\<Application>\user` folder double click the `jacc.bat` file
- In the `<JIS9.2.3>\jacadafiles\src\appls\<Application>\server\user` folder double click the jacc.bat file
- Repeat the previous two steps for each library `<JIS9.2.3>\jacadafiles\src\appls\<Library>`

Create a Runtime Installation

Java client

- Open the `jacadasv.ini` file from the previous version in the `<JIS>\jacadafiles\classes` folder
- Copy the `[MakeJar]` section
- Open the `<JIS9.2.3>\jacadafiles\classes\jacadasv.ini` file and paste the `[MakeJar]` section into the file and save it.
- Open the application in the JIS ACE Designer
- Select Utility/Create Runtime Installation
  - Select Stand Alone Server for the default JIS server or Application server for the J2EE deployment
  - Create IzPack Installation
  - Select Runtime type
  - Select Full Runtime for the scope
  - Select Server Platform
  - Select the Default Installation Directory
  - Select any image files for the installation splash screen
  - Click Finish

XHTML client

- Open the application in the JIS ACE Designer
- Select Utility/Create Runtime Installation

- Select Stand Alone Server for the default JIS server or Application server for the J2EE deployment
- Create IzPack Installation
- Select Runtime type
- Select Full Runtime for the scope
- Select Server Platform
- Select the Default Installation Directory
- Select any image files for the installation splash screen
- Click Finish

Install the JIS Runtime installation

- Copy the contents of the `<JIS9.2.3>\appls\<Application>\install` folder to the server where the JIS runtime installation will be installed.
- Install the Azul server JRE
- Install the JIS server runtime. For windows, execute: `RuntimeInstallation.exe.` For all other server platforms use the following command to install the RuntimeInstallation:

```
java -jar RuntimeInstallation.jar
```

Test the Runtime

Perform the necessary steps to insure the integrity of the upgraded application.

General Data Protection Regulation

While JIS Interface Server does not collect or store any user personal information directly, the server may log personal information from the host application depending on the specific host application and server debug level.

At debug level 70 or higher, client access logging will be enabled and server / host interaction will also be logged. These logs may contain personal information depending on the host application and server transactions.

Setting the server debug level to 10 or lower will disable access logging between the client and the server and will also disable logging of server / host interaction.

Any personal information in the JIS Interface Server logs can be removed by deleting the server logs.

Retirement of Product Components

As of release 9.2.3 the following product components are retired:

- Java Client Applet Support

Java Client Applet Support

Although Software GmbH currently supports running JIS Java clients as applets, support for the Java Plugin has been dropped from Chrome, Firefox and Microsoft Edge. In addition, Oracle deprecated the Java Plugin in JDK 9, effectively limiting support for running JIS Java client applets and Java applets in general, to Java 8 on Internet Explorer 11.

Support for Java client applets will end with the end of support for Internet Explorer. Java Web Start support has been added as an alternative to running Java client applets. Please see "Open Web Start Support" on page 75 for details.

New Features Included in JIS 9.2.3

The following new features have been added for JIS 9.2.3:

- OpenJDK Support
- Open Web Start Support
- Command Line Import and Export
- Upgrade to Jetty 9.4.40
- Java Server Page Support in the Embedded Jetty Server
- Proxy host and port settings for the MakeJar Time Stamp Authority
- SSL Cipher suite and protocol settings for Java client ports
- HTTP Request and response buffer size settings
- HTTP header settings
- Disable HTTP methods
- Secure HTTP Cookies
- Pause at the end of jacc.bat

OpenJDK Support

Both the server and clients are now supported on OpenJDK, specifically Azul Zulu Java 8. The bundled JRE and tools are unchanged from the previous release: Java 1.8.0_102.

Open Web Start Support

With Oracle dropping support for Java Web Start in Java 9, support for Open Web Start has been included in JIS 9.2.4 to provide an alternate means for launching JIS Java client applications via JNLP. Open Web start only supports lauching Java applications and does not support launching Java Applets, so existing client application which extend Applet will have to be regenerated. Open Web Start is not browser dependent and should work for the forseeable future.

Starting Java Clients With Java Web Start / Open Web Start

When deploying to production, the runtime installation package provides an additional option to rely on Java Web Start technology by launching the application from the auto generated `<App>-jnlp.html` file.

In addition, once a JNLP file has been downloaded it can be used to launch the application from the user's desktop just by clicking it like any normal executable link.

Starting the JIS Adminstrator With Java Web Start

The JIS Administrator utility may also be run as a Java Web Start application. Use the following options to load the administrator as a Java Web Start application:

- From the Java client startup page click the "administrator" link
- In standalone server use the following URL:
  `http://localhost:8080/classes/admin.jnlp`
- In application server deployment with a war file deployed to Jetty running on localhost, use a URL similar to this example:
  `http://localhost:8080/TEST923/admin`

> **Note:** Because IIS does not support running the JNLPDownloadServlet or provide similar functionality, launching a Java client via JNLP is not supported for users running IIS instead of the embedded Jetty web server.

Command Line Import and Export

Command-line access to ACE now includes the ability to import and export JIS applications. The command-line export operation exports the full application, including all libraries and subapplications. The command-line import operation imports the full contents of the export directory. Selective export can be done

using the ACE GUI and subsequent imports of that export directory will import only those application components selected for export. Selective import from a full export can be done via the ACE GUI.

Command-line access to ACE enables automating the following operations::

- Generate Runtime
- Create Runtime Installation
- Import/Export
- Pack/Unpack

The command-line access mechanism allows external tools to run ACE in automatic mode, using an XML file that contains a list of operations to execute in ACE.

The XML file must be named buildapp.xml.

Place the buildapp.xml file in the folder from which you launch ACE (the folder containing the ACE executable, generally the JIS installation folder).

To launch ACE in automatic mode:

- Use the following command parameter: `-oREMOTE`
  For example, to launch JIS XHTML for 3270 in automatic mode, use the following command line:

```
ACE.EXE -lmp -oREMOTE
```

When running ACE with the -oREMOTE command-line parameter, the buildapp.xml is read and the operations are executed by ACE.

Running ACE in Automatic Mode

When ACE is launched in automatic mode, it creates a GUI. Additional windows, such as the Pack animation window, are also displayed. However, in automatic mode, the ACE GUI is disabled.

When all of the operations run successfully (without errors), no user intervention is required. So, for instance, the Generate Runtime process dialog still opens, but closes automatically when it is done. If, however, there is an error message, such as an alert about missing image files, then this message is shown, and the user needs to click *OK* to close it. For more information, see “Error Handling” on page 79.

Once a valid `buildapp.xml` file has been created and tested, the operations run in ACE without the need for user intervention.

Changes in the Behavior of ACE Operations

The following sections describe changes in the behavior of ACE operations when using automatic mode.

Changes in Generate Runtime

The Generate Runtime process is always carried out for the entire application. It is not possible to specify specific libraries or subapplications. To speed things up, you can use the option to compile only new and modified subapplications.

Changes in Create Runtime Installation with Wise

After creating a runtime installation with Wise, ACE asks whether to launch the newly-created installation. In automatic mode, this question is skipped, and the installation is not launched.

Changes in Export

When exporting an application, it is possible to select individual components (libraries and/or subapplications). In automatic mode, the entire application is exported.

Changes in Import

When importing an application, it is possible to select individual components (libraries and/or subapplications) from those exported. In automatic mode, all of the exported components are imported.

Changes in Pack

When packing an application, it is possible to select the libraries to pack, and to add additional files, but the other steps of the Pack Wizard are not supported. Thus, for example, it is not possible to specify a maximal file size, nor to skip input directories (such as skipping DDS files, installation files and configuration files). All the files are included in the package (including configuration files).

Changes in Unpack

When unpacking an application, existing files are automatically replaced.

The configuration files (the files asked about in the last step of the wizard) are unpacked according to the settings chosen the last time the wizard was run from ACE.

The following example demonstrates various operations, such as opening an application, generating a runtime, creating a runtime installation, and packing and unpacking an application:

Example 7. JIS 9.2.3 Sample buildapp.xml

```
<Ace>
 <OpenApplication Name="TEST1">
  <GenerateRuntime Type="Java;XHTML" Platform="Windows; Solaris;
                  OS390;AS400;AIX;Linux" NewAndModified="0" />
 </OpenApplication>

 <OpenApplication Name="TEST2">
  <GenerateRuntime Type="Java;XHTML"
           Platform="Windows;Solaris;Linux;AIX;AS400;OS390" />
  <CreateRuntimeInstallation DeploymentType="Standalone"
     Platform="Windows" Runtime="XHTML" InstallFileSize="1024">
   <Wise Launch="1" ExecuteFile="C:\Program Files\Wise
         InstallMaster Demo\wise32.exe"
         InstallationDirectory="C:\TEST2" ImageFile="">
   </Wise>
  </CreateRuntimeInstallation>

  <CreateRuntimeInstallation DeploymentType="J2EE"
                          Server="weblogic;tomcat;websphere">
   <Libraries List="MODELS"/>
   <AdditionalFiles>
    <File Name="c:\temp\debug_1.log" Target="\WEB-INF\Lib" />
    <File Name="text2" Target="\WEB-INF\classes" />
   </AdditionalFiles>
  </CreateRuntimeInstallation>
 </OpenApplication>

 <ExportApplication Name="TEST2" GeneralSettings="0"/>

 <ImportApplication Name="TEST2" Path="c:\JIS\ImportExport"
                   GeneralSettings="0" DeleteImported="1"/>

 <Pack File="c:\temp\packtest.jpk" Name="TEST2" Libraries= ";"
                                        AdditionalFiles=";" />
 <Unpack File="c:\temp\myapp.jpk" Type="Java"          Target="MYAPP"
         InputDirectories="DDS;SDF;Screens"
         OutputDirectories="MakeExe;Runtime;Install"
```

```
        IncludeExtraFiles="True"  Existing="Replace"
        ConfigurationFiles="All">
 </Unpack>
</Ace>
```

For the complete DTD, which describes all possible attributes, refer to the `Buildapp.DTD`.

Logging

When running ACE in automatic mode, the following log is created: `remote.log` in the ACE root folder. The log begins with the contents of the `buildapp.xml` file. The `remote.log` reports the progress of the operations listed in the `buildapp.xml` file. In addition, information is also logged to the standard logs created by the Generate Runtime, Import/Export and Pack/Unpack Wizards.

Error Handling

When there is an error in any of the operations in the file, processing will terminate immediately. This prevents the accumulation of several problems, one on top of the other. For instance, if generating the runtime fails, then creating a runtime installation might create an installation of the previous version.

The following table describes different types of errors that may occur, and how to handle them.

Table 5 - 17 : JIS 9.2.3 Command-Line ACE Error Resolution

Error Types	Reasons for Errors	What to Check
Syntax Errors	• Malformed XML (i.e. XML tags not closed) • Children tags appear outside of parent tags • Missing values in XML, missing attributes	• Verify the syntax of the XML • Verify that the XML conforms to the DTD.

Table 5 - 17 : JIS 9.2.3 Command-Line ACE Error Resolution

Error Types	Reasons for Errors	What to Check
Logical Errors	• Trying to generate a runtime that is not allowed for the application by the CDKey • Creating a runtime installation before the application was ever compiled	• Manually execute the same operations from the ACE UI.

We recommend performing the operations the first time manually, using ACE, in the same order as in the buildapp.xml, and verifying that they work properly, before using the automatic mode.

Limitations

The following known limitations exist:

- The feature is certified only for the following product flavors:
  - JIS XHTML for 5250
  - JIS Java for 5250
  - JIS XHTML for 3270
  - JIS Java for 3270
- The main window of ACE is still visible in automatic mode
- The operation *Create Standalone Runtime Installation* is only supported for *Wise* runtime installations.
- The operation *Create J2EE Runtime Installation* requires that you Generate Runtime for the application and all of its libraries at least once from within ACE. Otherwise, you get the following Perl error:

  ```
  "Error: Key 'libraries' not found in section 'program' at
  …\\perl\gen/HierarchyFile.pl line 69".
  ```

  Workaround:

  Generate the runtime in automatic mode, then edit the `<AppName>.ini` and add the following setting:

  ```
  [Program]
  Libraries=<semicolon separated list of libraries>;
  ```

  If you do not have any libraries, then the list should just contain the name of the application. For example, Libraries=TEST; for an application named TEST.
- Do not use comments (<!-This is a comment->) in the buildapp.xml file.

- When the packaging of the J2EE runtime installation fails, remote.log still reports successful completion.
- When using the command line interface, if you set the NewAndModified attribute to 1 the first time you generate runtime, the compilation fails. The first time you compile you must not use NewAndModified.

Upgrade to Jetty 9.4.40

The internal Jetty web server has been upgraded to version 9.4.40 in order to utilize the latest security and performance enhancements.

Java Server Page Support in the Embedded Jetty Server

In order to provide greater flexibility in applications started via JNLP, support for JSP processing has been added to the embedded Jetty server.

Applications started via JNLP do not have access to the application command line requiring any arguments to be passed via the JNLP file. This can be done dynamically by using a JSP to generate the JNLP file at runtime. JSP processing is disabled by default and can be enabled by adding a `JSPEnabled` property setting to the `[HTTP]`section of the `jacadasv.ini` file:

```
[HTTP]
JSPEnabled=1
```

Enabling JSP support allows the embedded Jetty to compile Java Server Pages at runtime, providing the ability to dynamically generate web pages.

Because applications started via Java Web start do not have access to the HTTP request or URL used to launch the application, it is not possible to pass parameters to the application via the URL query string. JSP support was added specifically to address this limitation. By invoking a JSP to dynamically generate the JNLP request used to start a JIS application, information from the HTTP request, the URL or other dynamically or statically generated information can be passed to the JIS application through the generated JNLP property or param elements.

Sample HTML (<appl>-jsp.html) and JSP (<appl>.jsp) launcher pages are generated for Java clients during runtime generation. The <appl>-jsp.html page can be used to invoke the <appl>.jsp page or the <appl>.jsp page can be invoked directly. Parameters can be passed on the URL via the query string or hard coded into the <appl>-jsp.html page. The application Applet class generated during runtime generation has also been modified to provide an example of how to retrieve the parameters passed to the application.

JSP Examples

In these examples, we use <appl> to refer to the application.

> **Note:** The <appl>-jsp.html, <appl>.jsp and Applet class files are created during runtime generation so you must regenerate the runtime for each of your applications after installing this patch

The skeleton base-jsp.html file contains an example `<iframe>` tag demonstrating how to pass static parameters on the JSP URL query string (see `JacadaFiles/jskeletons/html/base-jsp.html`):

```
<iframe
src="<appl>.jsp?PRIMARY_COLORS=Red&PRIMARY_COLORS=Blue&PRIMARY_COLORS=Gr
een&SECONDARY_COLORS=Cyan&SECONDARY_COLORS=Magenta&SECONDARY_COLORS=Yell
ow&FAVORITE_COLOR=Blue">
</iframe>
```

In this example, the following parameters are passed to the `<appl>.jsp` page on the query string:

```
PRIMARY_COLORS = Red, Blue, Green
SECONDARY_COLORS = Cyan, Magenta, Yellow
FAVORITE_COLOR = Blue
```

`PRIMARY_COLORS` and `SECONDARY_COLORS` are array values and `FAVORITE_COLOR` is a scalar value.

Browsing to a URL for an `<appl>-jsp.html` page with the `<iframe>` tag

show above, for example:

```
(http://localhost:8080/<appl>-jsp.html)
```

would cause the browser to send a request for `<appl>.jsp` with the query string shown above. The server would then compile the JSP and run it, passing the HTTP request and associated query string to the JSP. The JSP would then generate a JNLP request and send it to the server.

An alternative method for passing dynamic parameters to a JSP is to add the parameters in a query string appended to the URL submitted to the browser. For example, to bypass the <appl>-jsp.html page and invoke the JSP directly passing a user ID and password, the following URL could be used:

```
http://localhost:8080/<appl>.jsp?USER_ID=MyUserID&PASS=MyPass
```

The skeleton `base.jsp` file (see `JacadaFiles/jskeletons/jsp/base.jsp`) contains examples of how to retrieve information from the HTTP request (request URI, request URL, servlet path, code base, query string and parameters with known names), how to pass parameters as `<resources><property>` elements and how to pass parameters as `<applet-desc><param>` elements.

Continuing the first (static) example, the following code in the JSP would iterate over the HTTP request, inserting each of the parameters from the request parameter map into the generated JNLP request as <applet-desc><param> elements:

```
<%
                // Iterate over the request params and pass them as params.
                System.out.println("HTTP Request Params:");
                Map<String,String[]> parms = request.getParameterMap();
                for (String key:parms.keySet()) {
                    String[] vals = (String[])parms.get(key);
                    System.out.print("Key: " + key + " = ");
                    String sep = "";
                    for (String val:vals) {
                       System.out.print(sep + val);
                       sep = ", ";
                       out.println("<param name=\"" + key + "\" value=\""
                                   + val + "\"/>");
                    }
                    System.out.print("\n");
                }
%>
```

In the second (dynamic) example, the following code in the JSP would get the USER_ID and PASS parameters from the HTTP request:

```
        <%
              String userId = request.getParameter("USER_ID");
              String password = request.getParameter("PASS");
        %>
```

and add them to the JNLP request as <applet-desc><param> elements:

```
        <applet-desc name="$application" main-
class="appls.$application.user.Applet" width="1024" height="768">
            <!-- .... -->
            <!-- Pass the USER_ID and PASS parameters via params. -->
            <param name="USER_ID" value="<%= userId %>"/>
            <param name="PASS" value="<%= password %>"/>
        </applet-desc>
```

In addition, the JSP URL encodes the URL query string:

```
        <%
              String queryStr = request.getQueryString();
              String encodedQueryStr = null;
              if (queryStr != null && queryStr.length() > 0) {
                  // URL encode the query string.
```

```
                encodedQueryStr = URLEncoder.encode(queryStr);
            }
        %>
```

and includes it in the generated JNLP as an <applet-desc><param>:

```
        <applet-desc name="$application" main-
class="appls.$application.user.Applet" width="1024" height="768">
            <!-- .... -->
            <!-- Pass the encoded JSP query string to the applet via param -
->
            <param name="QUERYSTR" value="<%= encodedQueryStr %>"/>
        </applet-desc>
```

The JNLP request is processed by the server which sends the request application back to the browser where it is launched using Java Web Start.

The Applet class (see `JacadaFiles/src/appls/<appl>/user/Applet.java)` generated for applications has been modified to retrieve the `QUERYSTR` parameter and pass it to the `getQueryParms` method which decodes and parses it for parameters, inserting them into the `htmlQueryParms` Hashtable, for use by the application.

Proxy host and port settings for the MakeJar Time Stamp Authority

In some cases, due to network constraints, it may be necessary to specify a proxy to allow access to time stamping authority when signing jar files generated during creation of a runtime installation. The HTTP proxy settings allow the user to configure communication with the TSA via a proxy over HTTP or HTTPS.

```
[MakeJar]
httpProxyHost=<proxyHostHttpAddr>
httpProxyPort<proxyHostHttpPort>
httpsProxyHost=<proxyHostHttpsAddr>
httpsProxyPort=<proxyHostHttpsPort>
```

SSL Cipher suite and protocol settings for Java client ports

It is now possible to configure the SSL protocols and cipher suites enabled for communication between JIS Java clients and the JIS server ports. This setting does not affect the embedded Jetty web server HTTP communications which are managed with property settings in the HTTP section of the `jacadasv.ini`.

```
[GeneralParameters]
SupportedProtocols=TLSv1.3
```

```
SupportedCipherSuites=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
```

HTTP Request and response buffer size settings

In some cases, large amounts of data passed between a browser and the JIS server via HTTP headers may cause a buffer overflow resulting in a "`Header is too large`" error message in the server log. The HTTP request and response buffer sizes may now be adjusted with the following property settings:

```
[HTTP]
HttpRequestHeaderSize=
HttpResponseHeaderSize=
```

HTTP header settings

For security purposes, the following HTTP headers may be configured to be sent with all HTTP response from the JIS web server:

- Cache-Control
- Content-Security
- Strict-Transport-Security
- X-Content-Type-Options
- X-Frame-Options
- X-Xss-Protection

```
[HTTP]
CacheControl=<directive>
ContentSecurity=<directive>
StrictTransportSecurity=<directive>
XContentTypeOptions=<directive>
XFrameOptions=<directive>
XXssProtection=<directive>
```

Note: In the example above, `<directive>` is an appropriate directive for the header in question. No validation of the directive is done by the JIS server.

Cache-Control

The CacheControl property in the HTTP section can be used to configure an HTTP Cache-Control header for all HTTP responses sent from the JIS Jetty server. The value of the CacheControl property will be used as the value of the directive for the Cache-Control header. If the CacheControl property is unset, no Cache-Control header will be included in HTTP responses from the server.

```
[HTTP]
CacheControl=no-cache='Set-Cookie, Set-Cookie2', must-revalidate
```

The default value for the `CacheControl` property is unset.

> **Note:** This header may be overridden by the Pragma: no-cache setting sent with Xhtml pages generated at runtime.

Content-Security-Policy

The `ContentSecurityPolicy` property in the HTTP section can be used to configure an HTTP `Content-Security-Policy` header for all HTTP responses sent from the JIS Jetty server. The value of the `ContentSecurityPolicy` property will be used as the value of the directive for the `Content-Security-Policy` header. If the `ContentSecurityPolicy` property is unset, no `Content-Security-Policy` header will be included in HTTP responses from the server.

```
[HTTP]
ContentSecurityPolicy=script-src 'self' 'unsafe-inline' 'unsafe-eval';
style-src 'self' 'unsafe-inline'
```

The default value for the `ContentSecurityPolicy` property is unset.

Strict-Transport-Security

The `StrictTransportSecurity` property in the `[HTTP]` section can be used to configure an HTTP `Strict-Transport-Security` header for all HTTP responses sent from the JIS Jetty server. The value of the `StrictTransportSecurity` property will be used as the value of the directive for the `Strict-Transport-Security` header. If the `StrictTransportSecurity` property is unset, no `Strict-Transport-Security` header will be included in HTTP responses from the server.

```
[HTTP]
StrictTransportSecurity=max-age=31536000;includeSubDomains
```

The default value for the `StrictTransportSecurity` property is unset.

X-Content-Type-Options

The `XContentTypeOptions` property in the `[HTTP]` section can be used to configure an HTTP `X-Content-Type-Options` header for all HTTP responses sent from the JIS Jetty server. The value of the `XContentTypeOptions` property will be used as the value of the directive for the `X-Content-Type-Options` header. If the `XContentTypeOptions` property is unset, no `X-Content-Type-Options` header will be included in HTTP responses from the server.

```
[HTTP]
XContentTypeOptions=nosniff
```

The default value for the `XContentTypeOptions` property is unset.

X-Frame-Options

The `XFrameOption` property in the HTTP section can be used to configure an HTTP `X-Frame-Options` header for all HTTP responses sent from the JIS Jetty server. The value of the `XFrameOption` property will be used as the value of the directive for the `X-Frame-Options` header. If the `XFrameOption` property is unset, no `X-Frame-Options` header will be included in HTTP responses from the server.

```
[HTTP]
XFrameOptions=SAMEORIGIN
```

The default value for the `XFrameOption` property is unset.

> **Note:** The `[Xhtml]XFrameOptions` setting will override the `[HTTP]XFrameOptions` setting for Xhtml responses sent by the server.

X-Xss-Protection

The `XXssProtection` property in the `[HTTP]` section can be used to configure an HTTP `X-Xss-Protection` header for all HTTP responses sent from the JIS Jetty server. The value of the `XXssProtection` property will be used as the value of the directive for the `X-Xss-Protection` header. If the `XXssProtection` property is unset, no `X-Xss-Protection` header will be included in HTTP responses from the server.

```
[HTTP]
XXssProtection=1; mode=block
```

The default value for the `XXssProtection` property is unset.

> **Note:** The X-Xss-Protection header is not supported by all browsers and will have no effect on those browsers that do not support it.

Disable HTTP methods

The `DisableHttpMethods` and `DisabledHTTPMethods` properties in the `[HTTP]` section can be used to disable certain methods supported by the embedded Jetty HTTP server. If the `DisableHttpMethods` property is enabled:

```
[HTTP]
DisableHttpMethods=1
DisabledHttpMethods=HEAD,OPTIONS,TRACE
```

the HTTP methods listed in the `DisabledHttpMethods` property will be disabled by the HTTP server. Browsers submitting requests which attempt to invoke any of the disabled methods will receive a "`403 Forbidden`" response.

The `DisableHttpMethods` property is enabled by default.

The `DisabledHttpMethods` property contains the following method names by default: "`HEAD, OPTIONS, TRACE`".

Secure HTTP Cookies

The `SecureCookies` property in the `[HTTP]` section can be used to control the addition of the `SECURE` and `HTTPOnly` attributes to HTTP cookies sent to the browswer. If the `SecureCookies` property is enabled, the `SECURE` and `HTTPOnly` attributes will be added to the cookie.

The SecureCookies property is enabled by default.

```
[HTTP]
SecureCookies=1
```

Pause at the end of jacc.bat

The jacc.bat file generated during application runtime generation now includes a pause command at the end of the file allowing users to see the results when running it from the Windows File Explorer without having to edit the file.

Detailed Description of Fixes Included in JIS 9.2.3

The following fixes are included in JIS 9.2.3:

- JIS-2029 / 1292251 - 9.2.2 - Compile Error after upgrading
  JIS-2033 - Application JNLP file will not run with jar files signed with any other certificate than the SAG certificate
  JIS-2044 - Add support for enabling JSP processing to embedded Jetty server
  JIS-2114 - Fix parameter substitution broken by JIS-2044
- JIS-2047 / 1302354 - Starting Server Error: java.lang.NoClassDefFoundError: jnlp/sample/servlet/JnlpDownloadServlet
  JIS-2048 - Include JacadaFiles/utils/jnlp in JIS XHTML installations
- JIS-2059 / 5304250 - The X and Y position of controls are different from v9.2 and v9.2.2
  JIS-2080 - Fix for backward compatibility of checkbox location with JIS 9.2 & 9.2.1
- JIS-2072 / 5309216 - javax.servlet.UnavailableException: Init parameter 'proxyTo' is required.
- JIS-2082 / 5312165 - Compile in JIS 9.2.2 not including the proper path names resulting in an HTTP Error 404
  JIS-2084 - Enable path compaction in JIS Jetty
- JIS-2090 / 5317186 - Case related to Incidents 5296922 and 5307096
- JIS-2106 / 5325425 - Not able to clone CheckBox in our application (From 9.2.0 to 9.2.2 Upgrade)
  JIS-2108 - Not able to clone CheckBox in our application (From 9.2.0 to 9.2.2 Upgrade)
  JIS-2132 - Fix the names of hidden subcomponents
- JIS-2123 / 5329638 - JIS 9.2.2 not showing input and output colors correctly
  JIS-2133 - "Prevent replacement of ""-"" with "" -"" in server output."
- JIS-2130 / 5334808 - Space before Dash removed in multiple field line comboBox or window title (post fix 5325425)
  JIS-2131 - "Fix for window title and combo-box entries with "" -"" in them getting changed to ""-"""
- JIS-2136 / 5339002 - 2 combo boxes working in JIS 9.2.1 but not JIS 9.2.2
- JIS-2156 / 5351852 - Problem with JavaClientSSLEnabled
- JIS-2163 / 1358933 - Need help with [MakeJar] section of the jacadasv.ini
  JIS-2164 - Add support for proxy host and port properties to the makejar section of the jacadasv.ini file
- JIS-2182 / 5365887 - Fortify scan issues
  JIS-2188 - Fixes for fortify scan issues
- JIS-2240 / 1414399 - Run Inside the browser is not working correctly
  JIS-2242 - Modify base.html template to make applet frame the size of the browser window.
- JIS-2244 - Update signing certificate for 2020-2022

- JIS-2252 / 5416624 - Disable HTTP OPTIONS Method
  JIS-2264 - Add property setting to allow disabling HTTP methods
- JIS-2260 / 5423256 - JIS jetty scan QID 11827 HTTP Security Header not detected
  JIS-2331 - Add config options for HTTP response headers
- JIS-2261 / 5423985 - Lack of Response Headers: HSTS
  JIS-2331 - Add config options for HTTP response headers
- JIS-2266 / 1428350 - Timing for transactions
  JIS-2267 - Add logging to make timing of HTTP requests easier
- JIS-2281 / 5428809 - BBI Blackduck scanning-license and security issues
  JIS-2329 - Upgrade embedded Jetty to Jetty-9.4.40
- JIS-2282 / 5427697 - Various GUI controls are not functioning in Chrome
  JIS-2321 - Fix Xhmtl client date control problem on IE
- JIS-2283 / 1432003 - Prototype.js receiving error in chrome
- JIS-2285 / ZT-???? - New Vulnerability Items from Client
  JIS-2309 - Upgrade embedded Jetty to Jetty-9.4.38
- JIS-2310 / 5439014 - JIS Crashed on AMVC BBI screen
  JIS-2311 - Fix NullPointerException
- JIS-2324 / 1445798 - X-Content-Type-Options header for JIS
  JIS-2331 - Add config options for HTTP response headers
- JIS-2325 / 1445800 - Security Policy Header
  JIS-2331 - Add config options for HTTP response headers
- JIS-2326 / 1445799 - X_XSS Protection Response Header
  JIS-2331 - Add config options for HTTP response headers

ACE

JIS-2163 / 1358933 - Need help with [MakeJar] section of the jacadasv.ini
JIS-2164 - Add support for proxy host and port properties to the makejar section of the jacadasv.ini file

Propertysettings were added to the `[MakeJar]` section of the `jacadasv.ini` file to allow configuration of a proxy host and port for communication with TSA through a proxy.

Java client

JIS-2029 / 1292251 - 9.2.2 - Compile Error after upgrading
JIS-2033 - Application JNLP file will not run with jar files signed with any other certificate than the SAG certificate
JIS-2044 - Add support for enabling JSP processing to embedded Jetty server
JIS-2114 - Fix parameter substitution broken by JIS-2044

The JNLP specification requires all jar files referenced by a JNLP file to be signed with the same signing certificate. This would require the JIS clbase jar file and the customer generated application jar file to be signed with the same certificate. To resolve this problem, the JNLP file generated for a JIS application has been split into two parts, an application specific file and a clbase file. This allows the application code to be signed with a different signing certificate than the JIS clbase client jar file.

JIS-2240 / 1414399 - Run Inside the browser is not working correctly
JIS-2242 - Modify base.html template to make applet frame the size of the browser window.

The JIS Java client base.html skeleton file was modified to use 100% of the browser frame it is launched in when `RunInsideBrowser=TRUE`.

XHTML Client

JIS-2059 / 5304250 - The X and Y position of controls are different from v9.2 and v9.2.2
JIS-2080 - Fix for backward compatibility of checkbox location with JIS 9.2 & 9.2.1

A backward compatibility problem with the `getLocation()` method of the CheckBox control was fixed.

JIS-2282 / 5427697 - Various GUI controls are not functioning in Chrome
JIS-2321 - Fix Xhmtl client date control problem on IE

A number of browser compatibility problems were fixed:

- Group boxes not sized correctly
- Tab control not letting user select subsequent tabs
- Right click to view options on a table not working in IE
- Calendar control not positioned incorrectly
- List table pop-up positioned incorrectly
- Radio group button labels not selectable
- Dynamic Checkbox only displays label (no checkbox)

JIS-2283 / 1432003 - Prototype.js receiving error in chrome

A browser compatibility problem was fixed.

Installation

JIS-2244 - Update signing certificate for 2020-2022

The certificate used to sign JIS Interface Server code and jar files was updated.

Server

JIS-2047 / 1302354 - Starting Server Error: java.lang.NoClassDefFoundError: jnlp/sample/servlet/JnlpDownloadServlet
JIS-2048 - Include JacadaFiles/utils/jnlp in JIS XHTML installations

As of the release of JIS 9.2.2, the jnlp-servlet.jar file is required to start the JIS server. It is now installed in all configurations.

JIS-2072 / 5309216 - javax.servlet.UnavailableException: Init parameter 'proxyTo' is required.

A Jetty server configuration problem was fixed.

JIS-2082 / 5312165 - Compile in JIS 9.2.2 not including the proper path names resulting in an HTTP Error 404
JIS-2084 - Enable path compaction in JIS Jetty

A change in behavior introduced with the latest Jetty upgrade was resolved with a configuration change. Previous versions of Jetty allowed // in URLs while newer versions do not. Paths are now compacted.

JIS-2090 / 5317186 - Case related to Incidents 5296922 and 5307096

Fixes related to Java Web Start.

JIS-2106 / 5325425 - Not able to clone CheckBox in our application (From 9.2.0 to 9.2.2 Upgrade)
JIS-2108 - Not able to clone CheckBox in our application (From 9.2.0 to 9.2.2 Upgrade)
JIS-2132 - Fix the names of hidden subcomponents

A problem with Xhtml client CheckBox controls was fixed.

JIS-2123 / 5329638 - JIS 9.2.2 not showing input and output colors correctly
JIS-2133 - "Prevent replacement of ""-"" with "" -"" in server output."

A problem with Xhtml client input and output field colors was fixed.

JIS-2130 / 5334808 - Space before Dash removed in multiple field line comboBox or window title (post fix 5325425)
JIS-2131 - "Fix for window title and combo-box entries with "" - "" in them getting changed to ""-""""

A regression in ComboBox behavior was fixed.

JIS-2136 / 5339002 - 2 combo boxes working in JIS 9.2.1 but not JIS 9.2.2

A regression in ComboBox behavior was fixed.

JIS-2156 / 5351852 - Problem with JavaClientSSLEnabled

A problem with the JavaClientSslOnly configuration was fixed.

JIS-2182 / 5365887 - Fortify scan issues
JIS-2188 - Fixes for fortify scan issues

Security issues flagged by a Fortify scan were fixed.

JIS-2252 / 5416624 - Disable HTTP OPTIONS Method
JIS-2264 - Add property setting to allow disabling HTTP methods

The `[HTTP] DisableHttpOptions` and `DisabledHttpMethods` properties were added to allow users to disable certain HTTP methods. See “Disable HTTP methods” on page 93.

JIS-2260 / 5423256 - JIS jetty scan QID 11827 HTTP Security Header not detected
JIS-2331 - Add config options for HTTP response headers

The `[HTTP] XFrameOptions` property was added to allow user to configure the HTTP `X-Frame-Options` header for all server responses. See “X-Frame-Options” on page 87.

JIS-2261 / 5423985 - Lack of Response Headers: HSTS
JIS-2331 - Add config options for HTTP response headers

The `[HTTP] StrictTransportSecurity` property was added to allow user to configure the HTTP `Strict-Transport-Security` header for all server responses. See "Strict-Transport-Security" on page 91.

JIS-2266 / 1428350 - Timing for transactions
JIS-2267 - Add logging to make timing of HTTP requests easier

Server logging was enhanced to enable timing of HTTP request/response transactions easier. When the JIS server is run at debug level 5 or higher, messages of the form:

```
Action handling total time:
```

are logged in the server log for each HTTP request.

JIS-2281 / 5428809 - BBI Blackduck scanning-license and security issues
JIS-2329 - Upgrade embedded Jetty to Jetty-9.4.40

The embedded Jetty server was upgraded to version 9.4.40 to resolve issues flagged by a security scans.

JIS-2285 - New Vulnerability Items from Client
JIS-2309 - Upgrade embedded Jetty to Jetty-9.4.38

The embedded Jetty server was upgraded to version 9.4.38 to resolve issues flagged by a security scans.

JIS-2310 / 5439014 - JIS Crashed on AMVC BBI screen
JIS-2311 - Fix NullPointerException

A NullPointerException resulting from an uncheck object reference was fixed.

JIS-2324 / 1445798 - X-Content-Type-Options header for JIS
JIS-2331 - Add config options for HTTP response headers

The `[HTTP] XContentTypeOptions` property was added to allow user to configure the HTTP `X-Content-Type` header for all server responses. See "X-Content-Type-Options" on page 87.

JIS-2325 / 1445800 - Security Policy Header
JIS-2331 - Add config options for HTTP response headers

The `[HTTP] ContentSecurityPolicy` property was added to allow user to configure the HTTP `Content-Security-Policy` header for all server responses. See "Content-Security-Policy" on page 86.

JIS-2326 / 1445799 - X_XSS Protection Response Header
JIS-2331 - Add config options for HTTP response headers

The `[HTTP] XXssProtection` property was added to allow user to configure the HTTP `X-Xss-Protection` header for all server responses. See "X-Xss-Protection" on page 87.

Limitations of JIS 9.2.3

The following are known limitations in JIS 9.2.3:

- JIS-2046 / 5298807 - Java clients started via Java Web Start can't find ports file
- JIS-2157 / 1356981 - JIS Administrator not showing ini settings
- JIS-2218 / 5389111 - POST Data in Worker Processes
- JIS-2221 / 1237366 - Add support for server / host SSL protocol and cipher suite settings
- JIS-2238 / 1298595 - IzPack deployment does not update an existing jacadasv.ini when deploying multiple applications to the same server
- JIS-2296 / 1362003 - Fix missing table row buttons
- JIS-2297 / 1347703 - Fix problem with modified JIS application JNLP files
- JIS-2262 / ZD-???? - Session ID in the clear

JIS-2046 / 5298807 - Java clients started via Java Web Start can't find ports file

Java clients launched via JNLP start searching for the JIS server ports file in the `classes` directory of the runtime installation. The ports file is typically installed in the root directory of the runtime installation causing the client to fail to find the ports file and fall back to the default ports. If the server is using non-default ports, the client will fail to connect to the server. The ports file can be copied to the classes directory as a workaround.

JIS-2157 / 1356981 - JIS Administrator not showing ini settings

The `jrodefaults.ini` file is not included in the runtime installation, causing the JIS Administrator to fail to show server ini settings. The workaround is to manually copy the `jrodefaults.ini` file from the development machine to the runtime deployment.

JIS-2218 / 5389111 - POST Data in Worker Processes

Worker processes lose access to XML POST data sent to the root node in a multi-process, scaled server configuration. This problem does not occur in a single process configuration. As a workaround, XML POST data can be sent directly to the worker processes in a multi-process, scaled server configuration

JIS-2221 / 1237366 - Add support for server / host SSL protocol and cipher suite settings

It is not currently possible to configure the SSL protocols or cipher suites enabled by the server for SSL/TLS server / host connections. The enabled protocols and cipher suites can be configured by setting the appropriate Java properties on the JVM command line.

JIS-2238 / 1298595 - IzPack deployment does not update an existing jacadasv.ini when deploying multiple applications to the same server

When multiple runtime installations are installed to the same server, an entry for only the first application is created in the `[Applications]`section of the `jacadasv.ini` file. Entries for subsequent applications, must be added manually.

JIS-2296 / 1362003 - Fix missing table row buttons

Table row buttons visible in the ACE designer and Java client are not shown in Xhtml clients.

JIS-2297 / 1347703 - Fix problem with modified JIS application JNLP files

If the JNLP file for an application that is already cached on the client is modified on the server, the client will fail to load with an EOF exception the next time the client tries to load the application. This can be worked around by restoring the original JNLP file on the server or by deleting the cached application on the client.

JIS-2262 - Session ID in the clear

The JIS Xhtml client makes use of the URL to pass the JIS client session ID between the client and the server. The session ID is encrypted when communicating over HTTPS but is passed in the clear when communicating over HTTP. The session ID can be viewed in the browser URL bar regardless of the communication protocol.

JIS Interface Server 9.2.2 Release Notes

Installation & Upgrade Information

Supported Platforms

JIS Interface Server 9.2.2 is supported on the following platforms:

- Windows Server 2008 Standard and Enterprise (32-bit)
- Windows Server 2008 Standard and Enterprise (64-bit)
- Windows Server 2012 Standard and Enterprise (64-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (32-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (64-bit)
- Windows 10 Professional, Ultimate and Enterprise Edition (64-bit)
- Solaris SPARC 11 (64-bit)
- AIX 7.1 Power (64-bit)
- AIX 7.2 Power (64-bit)
- Red Hat Enterprise Linux 6 for x86 (64-bit)
- Red Hat Enterprise Linux 7 for x86 (64-bit)

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for Java versions, browser versions and application server versions supported by their respective vendors. Being so, when a vendor no longer supports a Java version, browser version or application server version, Software GmbH will discontinue supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows 7
- Windows 10

Clients

Java Clients

The Java Client has been tested on the following operating systems, browser and Java versions:

- Windows 7
- Windows 10

Table 6 - 18 : JIS 9.2.2 Java Client Supported Browser and Java Versions

Browser	JRE
IE	Oracle1.8.0
Firefox 52 ESR	Oracle1.8.0

Support for the Java Plugin (required to launch Java applets) has been dropped in Chrome 45, Firefox 53 and Microsoft Edge, effectively removing them from the list of browsers supported for Java client applets. Internet Explorer and the Firefox 52 ESR 32-bit release offer continued support for the Java Plugin. Users of any newer releases of Firefox or other browsers which do not support the Java Plugin will need to use Java Web Start to launch Java clients. See "Java Web Start Support" on page 104 for more details.

XHTML Clients

The XHTML client has been tested with the following operating system and browser versions:

Table 6 - 19 : JIS 9.2.2 XHTML Client Supported Operating Systems and Browsers

Operating Syste	Browser
Windows	IE 8, IE 11, Edge
Windows	Firefox
Windows	Chrome
Mac OS/X	Safari
iPad iOS	Safari Mobile

Standalone Server

The JIS standalone server has been tested in the following environments:

Table 6 - 20 : JIS 9.2.2 Standalone Server Supported Environments

Operating System	Java Version
Windows 2008	Oracle 1.8.0
Windows 2012	Oracle 1.8.0
Solaris 11	Oracle 1.8.0
Red Hat Linux AS6	Oracle 1.8.0
Red Hat Linux AS7	Oracle 1.8.0
AIX 7.1	IBM 1.8.0
AIX 7.2	IBM 1.8.0

Application Server Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 6 - 21 : JIS 9.2.2 XHTML Client Supported Runtime Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WebSphere 8.5.5.	IBM JRE 1.7	Windows 2003
Tomcat 8	Oracle JRE 1.7	Windows 2008

OS400 Components

The DDS compiler has been tested on the following operating systems:

- OS400 V6R1
- OS400 V7R1

Upgrade Instructions

Runtime Installation Upgrade Instructions

JIS 9.2.4 includes an updated version of the IzPack runtime installer. Due to changes in IzPack itself there are several behavior changes which require upgrading existing customer installation projects.

To migrate to the new project structure customer would need to:

1. Backup and delete their project specific `<JISRoot>\appls\<AppName>\install\install.xml` file.
2. Generate a new runtime installation.
3. Merge project specific changes made to their old `install.xml` file into the newly generated `install.xml` file

In addition, creating a new runtime installation in the designer will no longer trigger the runtime installation executable.

Retirement of Product Components

As of release 9.2.2 the following product components are retired:

- Java Client Applet Support

Java Client Applet Support

Although Software GmbH supports running JIS Java Clients as applets, support for the Java Plugin has been dropped in Chrome 45, Firefox 53 and Microsoft Edge. In addition, Oracle is planning on deprecating the Java Plugin in JDK 9, effectively limiting support for running JIS Java client applets and Java applets in general, for customers who wish to remain on current browser and Java versions, to those running on Internet Explorer and Firefox 52 ESR 32-bit. Java Web Start support has been added as an alternative to running Java client applets. Please see "Java Web Start Support" on page 104 for details.

New Features Included in JIS 9.2.2

The following new features have been added for JIS 9.2.2:

- Java 8 Support
- Upgrade to IzPack 5.0
- Java Web Start Support
- Upgrade to Jetty 9.4.2
- Automatic HTTP to HTTPS Redirection
- Clickjacking Protection
- Optional Disabling of Caching of Table CSS Data
- Enable Compatibility Mode in Internet Explorer 11 Automatically

Java 8 Support

The bundled JRE and tools were upgraded to Java 1.8.0_102. Both the server and client are now supported on Java 8.

Upgrade to IzPack 5.0

The JIS runtime installer has been upgraded to IzPack 5.0. See "Runtime Installation Upgrade Instructions" on page 102 for instructions on upgrading existing installation projects.

Java Web Start Support

With the major browsers dropping support for Java applet technology, Java Web Start support has been included in JIS 9.2.4 to provide an alternate means for launching JIS Java client applications. Java Web Start is not browser dependent and should work for the forseeable future.

One of the requirements of Java WebStart is that it works only with signed Jar files and not with individual class files and resources. Therefore in order to use it, developers have to package their applications into Jar files and sign them. This process is now implemented by the IzPack runtime installation.

IE, Edge and Firefox automatically launch a WebStart application when provided with its JNLP file. Unfortunately, Chrome recognizes the JNLP file as a "Dangerous Attachment" and does not launch it before the user confirms a "Keep/Discard" warning message. This issue has been documented in https://bugs.chromium.org/p/chromium/issues/detail?id=518170 therefore we cannot declare Chrome support at the moment.

During development of the application, developers can still use Java Applet technology using one of the following methods:

- IE11 or lower + Java Plugin
- The AppletViewer tool provided with the Java JRE

Starting Java Clients With Java Web Start

When deploying to production, the runtime installation package provides an additional option to rely on Java WebStart technology by launching the application from the auto generated `<App>-jnlp.html` file

In addition, once a JNLP file has been downloaded it can be used to launch the application from the user's desktop just by clicking it like any normal executable link.

Starting the JIS Adminstrator With Java Web Start

The JIS Administrator utility may also be run as a Java Web Start application. Use the following options to load the administrator as a Java WebStart application:

- From the Java client startup page click the "administrator" link
- In standalone server use the following URL: `http://localhost:8080/classes/admin.jnlp`
- In application server deployment with a war file deployed to Jetty running on localhost, use a URL similar to this example: `http://localhost:8080/TEST922/admin`

Upgrade to Jetty 9.4.2

The internal Jetty web server has been upgraded to version 9.4.2 in order to utilize the latest security and performance enhancements.

Automatic HTTP to HTTPS Redirection

The purpose of this enhancement is to enable running both the HTTP and HTTPS ports open but whenever a user connects to the HTTP port, automatically redirect the request to the HTTPS port.

This behavior is disabled by default. To enable this setting, configure the server to work with HTTPS, then set the following `jacadasv.ini` setting:

Example8. JIS 9.2.2 HTTP Redirection Properties

```
[Http]
RedirectHttpToHttps=1
```

Given a server configured to listen for HTTP requests on port 8080 and HTTPS requests on port 8443, connections to `http://localhost:8080` would automatically redirect the browser to `https://localhost:8443`.

Clickjacking Protection

Clickjacking (https://en.wikipedia.org/wiki/clickjacking) is an attack in which a JIS session can be maliciously embedded inside an invisible frame so that clicking one of the legitimate buttons on a web page generated by JIS would actually trigger an action provided by the invisible frame. One way to mitigate this attack is using the `X-Frame-Options` Http header (https://developer.mozilla.org/en-us/docs/web/http/x-frame-options) which can be configured to prevent the embedding of the JIS session inside a frame. The `XFrameOptions <AppName>.ini` setting can be used to enable the `X-Frame-Options header`:

Example9. JIS 9.2.2 XFrameOptions Properties

```
[Xhtml]
XFrameOptions=<directive>
```

One of the following X-Frame-Options directives may be used:

Table 6 - 22 : JIS 9.2.2 X-Frame-Options Directives

Directive	Description
DENY	The page cannot be displayed in a frame, regardless of the site attempting to do so.
SAMEORIGIN	The page can only be displayed in a frame on the same origin as the page itself.
ALLOW-FROM uri	The page can only be displayed in a frame on the specified origin.

The X-Frame-Options headers are only enforced on browsers which support them.

Optional Caching of Table CSS Styesheet Resources

In some cases, when using Internet Explorer, the browser can send a conditional request for table CSS stylesheet resources to the server even though it does not have those resources in its cache. If the server returns a 304 (Not Modified) response the resulting page will be displayed incorrectly.

By enabling the `AlwaysGenerateTableCSS` setting, the server is prevented from sending the 304 response. Instead, the server will always generate the table CSS resources. To enable this behavior, disabling caching of CSS stylesheet resources, set the following `<AppName>.ini` setting:

Example10. JIS 9.2.2 CSS Stylesheet Table Caching Properties

```
[xhtml]
AlwaysGenerateTableCSS=1
```

By default, this setting is disabled, enabling caching of CSS stylesheet resources.

Enable Compatibility Mode in Internet Explorer 11 Automatically

Recent versions of JIS support IE11 by treating it as equivalent to Firefox and Chrome. This approach works in most cases, however under certain conditions this causes rendering problems, for example with table controls.

Since solving these issues completely requires a major re-architecture of the server Html rendering code, we decided in the meantime, to provide an alternative by switching IE11 to work in compatibility mode automatically.

This code relies on the X-UA-Compatible HTTP header and meta tag as explained here https://msdn.microsoft.com/en-us/library/jj676915(v=vs.85).aspx to instruct IE11 to switch IE11 to compatibility mode automatically.

The implementation of this fix requires several steps:

1. Identify IE11 + compatibility settings and use code compatible with old IE releases on the server side.
2. Write the X-UA-Compatible Http header to the page response with the value "IE=EmulateIE7" which seems to provide the best compatibility result.

The end result is that the JIS session can be launched from IE11 and it will automatically switch to compatibility mode without any user action.

To activate this mode in runtime requires the following settings:

1. Add a new ini setting to trigger the new behavior:
   `<AppName>.ini [Xhtml] UseCompatibilityModeForIE11=1`
2. Add the following meta tag to the launcher Html inside the <head> tag:

```
<html>
  <head>
    <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7">
    .
    .
    .
  <head>
<html>
```

Detailed Description of Fixes Included in JIS 9.2.2

The following fixes are included in JIS 9.2.2:

- JIS-1607 / 5140355 - Excess Server Logs
  JIS-1627 - Eliminate Session Dump when calling TotalExit
- JIS-1626 / 5143850 - Screen Flickering Issue for later 1.7.0_xx JRE versions
  JIS-1631 - Prevent load balancer from caching Java client HTTP requests
- JIS-1613 / 5140988 - MessageBox from Client, title: Server not responding
  JIS-1632 - Fix focus related Java client deadlock
- JIS-1634 - Java Client Help-->About dialog contains garbage char
- JIS-1635 - Administrator runtime settings XHTML section contains empty checkbox

- JIS-1639 - Update code signing certificate
- JIS-1648 / 5147214 - Problem with Keyboard Buffering
  JIS-1649 - Keyboard buffering enters an infinite loop
- JIS-1650 - KeepAlive requests are not sent by the client
- JIS-1651 / 5147560 - Server Disconnect on Text Entry screen
  JIS-1652 - Problem sending data when a java.awt.TextArea is the focused control
- JIS-1655 / 5148126 - Using Wise Installer to Create Runtime Installation
  JIS-1656 - Wise installer not launched by runtime installation wizard
- JIS-1661 - XSLT and JavaScript refactoring
- JIS-1669 - Upgrade to Jetty 9
- JIS-1671 / 5153539 - Maps Displayed After Natural Terminal Non Conversational Mode Activated
  JIS-1694 - Implement page loading indicator
- JIS-1678 - Use Servlet 3.0 AsyncContext in XHTML
- JIS-1682 / 5155150 - Upgrading to v9.21.- Javascript error
  JIS-1683 - Prototype.js causing JavaScript error with IE8
- JIS-1687 / 5155831 - Status of Java 8 certification
  JIS-1691 - Java 8 certification
- JIS-1689 - Use AsyncContext in HTTP proxy servlet
- JIS-1690 / 5155863 - Error popup during runtime - Couldn't create window
  JIS-1692 - Dynamically allocate components per subapplication
- JIS-1693 / 5150037 - Date control issue
  JIS-1711 - Date control fixes
- JIS-1701 - Compress response to client
- JIS-1704 / 1093683 - Password/hidden fields being logged.
  JIS-1707 - Password information printed to server log
- JIS-1708 / 5158088 - JIS service status on JIS server
  JIS-1740 - Do something about the missing -Xrs setting
- JIS-1719 / 5161179 - Formats.ini file not working after ugrade of JIS software from 9.2 to 9.2.1
  JIS-1726 - Fix dictionary upper case and separator symbols
- JIS-1721 / 5162246 - POODLE Security Vulnerability Breaks SSLv3 Secure Browsing
  JIS-1730 - Fix poodle security vulnerability
- JIS-1727 / 5162398 - Application causing delay and loading on JITGUI
  JIS-1744 - Fix emulator logic related to screen changes
- JIS-1737 - Java 8 support
- JIS-1746 / 5166996 - Print margin defaults changed with JIS 9.2.1
  JIS-1748 - Specify default page margins for print gui and print host screen
- JIS-1753 - HTTP to HTTPS redirection
- JIS-1754 / 5170343 - Delta - Need to inject <!DOCTYPE html> at the beginning of our page

JIS-1763 - Add support for the HTML5 DOCTYPE and remove XML declaration

- JIS-1760 / 5170962 - JIS application does not execute the prompting when prompt is issued from a table field
  JIS-1761 - Prompt button inside a table does not work
- JIS-1770 - Support for runtime installation Windows uninstall panel
- JIS-1778 / 5175923 - JIS 9.2.1 XHTML - View Host Screen
  JIS-1783 - Remove code duplication between view host and printing buttons
- JIS-1780 / 5175421 - Radio Group issues
  JIS-1782 - Fix radio button duplicate label on IE
- JIS-1800 / 5180922 - Look of JacadaList is Corrupted in IE11
  JIS-1835 - Adapt table and folded table to HTML5 rendering
- JIS-1802 / 5180721 - Many to one Screen issue
  JIS-1808 - Many To One - do not collect messages from other dependents
- JIS-1803 / 5182228 - JIS 9.2.1 getPostDate() API
  JIS-1804 - Change the name attribute from txt to txtarea for text area in non-IE browsers
- JIS-1810 / 5183374 - Slow transactions when connected to secure host port
  JIS-1814 - Specify TcpNoDelay when connecting to SSL server
- JIS-1811 / 5185330 - IP redirect
  JIS-1815 - BindIPAddressForHTTPClient setting is broken
- JIS-1821 / 5184220 - Behavior In IE11 different concerning the display of the page when submitting it
  JIS-1866 - Switching to IE11 compatibility mode automatically
- JIS-1826 - Fix JITGUI on IE11 with HTML5 Doctype
- JIS-1827 - Add FOCUS filter to track the Java client focus behavior
- JIS-1829 / 5189161 - Jetty URL case senitivity and ETag support
  JIS-1838 - Enable Etag response header to improve caching
- JIS-1846 / 5192404 - Package process for JIS - target root directory coming out in all caps
  JIS-1847 - Application Server deployment - fix case of RootDir parameter
- JIS-1848 / 5192870 - GEM - Missing XHTMLCSS Stylesheet Resource File - Causing "Expanding Tables"
  JIS-1852 - Table CSS refresh problem
- JIS-1863 / 1107543 - Issue with scroll bars on tables
  JIS-1870 - Subapplication specific screen system trigger methods are not invoked
- JIS-1888 / 5210202 - Unable to start application using an obfuscated JettyKeyStore password
- JIS-1893 - SSL socket cannot read obfuscated keystore password
- JIS-1895 / 5211802 - Converting from java aplets to XHTML - norwegian characters
  JIS-1906 - Use UTF8 encoding when converting the static XML file to a DOM
- JIS-1904 - Update runtime installation to IzPack 5.0

- JIS-1912 / 5218237 - Clickjacking protection
  JIS-1915 - Add support for the X-Frame-Options HTTP header to prevent clickjacking
- JIS-1938 - Change click event behavior for date picker
- JIS-1942 - Java Web Start support
- JIS-1950 - Web Start support
- JIS-1953 - 2016 code signing certificate
- JIS-1966 / 5244901 - Printing Issue
  JIS-1968 - Improve printing of image buttons
- JIS-1973 / 5248661 - Multiple JIS clients continually freeze
  JIS-1974 - Reduce PS items log verbosity
- JIS-1977 / 5251062 - Jetty web server configuration question
  JIS-1978 - Disbable unused HTTP methods and protect server identity
- JIS-1993 - Load the JIS administrator as a Java Web Start application
- JIS-1994 - Wize runtime installation - JIS server failed to start

ACE

JIS-1655 / 5148126 - Using Wise Installer to Create Runtime Installation
JIS-1656 - Wise installer not launched by runtime installation wizard

A bug which caused the IzPack installer to be launched even though the Wise installer was selected in the Create Runtime Wizard has been fixed.

Java client

JIS-1626 / 5143850 - Screen Flickering Issue for later 1.7.0_xx JRE versions
JIS-1631 - Prevent load balancer from caching Java client HTTP requests

Caching of Java client HTTP requests has been disabled to prevent load balancers to cause screen flickering for Java clients running on some Java versions.

JIS-1613 / 5140988 - MessageBox from Client, title: Server not responding
JIS-1632 - Fix focus related Java client deadlock

A deadlock caused by not using the event dispatch thread while removing components from a window during clean up has been fixed.

JIS-1634 - Java Client Help-->About dialog contains garbage char

The copyright symbol in the Java Client Help-->About dialog contained a garbage character. This has been fixed.

JIS-1648 / 5147214 - Problem with Keyboard Buffering
JIS-1649 - Keyboard buffering enters an infinite loop

A problem causing buffered key events and tab events generated in order to skip to the next field were being recorded causing an infinite playback loop has been fixed.

JIS-1651 / 5147560 - Server Disconnect on Text Entry screen
JIS-1652 - Problem sending data when a java.awt.TextArea is the focused control

Using a java.awt.TextArea from a Java extension and sending data to the server while the textArea component was in focus, terminated the session with the following exception "java.lang.ClassCastException: java.awt.TextArea cannot be cast to cst.gwt.GUIControlInterface".

This has been fixed.

JIS-1687 / 5155831 - Status of Java 8 certification
JIS-1691 - Java 8 certification

JIS Interface Server 9.2.2 is certified to run on Java 8.

JIS-1737 - Java 8 support

Modifications required to support building and running on Java 8 while maintaining backward compatibility with Java 7.

JIS-1746 / 5166996 - Print margin defaults changed with JIS 9.2.1 JIS-1748 - Specify default page margins for print gui and print host screen

Users can now specify the default page margins for print GUI and print host screen using the `PrintPageMargins` Applet parameter.

The "`PrintPageMargins`" parameter is a string value composed of 4 decimal values separated by comma, representing the left, top, right and bottom page margins correspondingly measured in inches.

For example, specifying:

```
<PARAM name="PrintPageMargins" value="0.75,0.5,0.5,0.5">
```

in a Java client HTML page, sets the left margin to 0.75 inch and the other margins to 0.5 inch for all Java client sessions started from that HTML page.

JIS-1827 - Add FOCUS filter to track the Java client focus behavior

If the `FOCUS` filter is enabled, a stack trace is logged on `GUICSTPanel focusGained`, `GUICSTPanel focusLost` and `component updateOnFocus` events.

The `FOCUS` filter can be enabled by adding the `-fFOCUS` command line option to the command line in the JIS server startup script, by setting the `RtDebugFilters` property in the `jacadasv.ini` file or at runtime by entering the `addfilter FOCUS` command in the server command window.

JIS-1863 / 1107543 - Issue with scroll bars on tables JIS-1870 - Subapplication specific screen system trigger methods are not invoked

A regression which caused subapplication specific trigger methods for the screen receiver, such as `PageUp`, `PageDown`, `GetToBottomOfList`, `GetToTopOfList`, etc. not to be invoked has been fixed.

JIS-1942 - Java Web Start support JIS-1950 - Web Start support

JIS 9.2.4 includes support for running Java clients via Java Web Start. See “Java Web Start Support” on page 104 for a detailed description.

JIS-1966 / 5244901 - Printing Issue
JIS-1968 - Improve printing of image buttons

Most of the GIF button images provided now print with a transparent background.

JIS-1993 - Load the JIS administrator as a Java Web Start application

The JIS Adminstrator utility may be loaded as a Java Web Start application. See "Java Web Start Support" on page 104for a detailed description.

XHTML Client

JIS-1607 / 5140355 - Delta - Excess Server Logs
JIS-1627 - Eliminate Session Dump when calling TotalExit

In XHTML, placing the `DoMethod TotalExit` after `MoveAccordingToHost` to terminate the session resulted in a session dump during the generation of the response. The session dump has been eliminated but an exception is still written to the server log.

JIS-1650 - KeepAlive requests are not sent by the client

A regression in JIS 9.2.1 caused the XHTML client to stop sending KeepAlive requests. This has been fixed.

JIS-1661 - XSLT and JavaScript refactoring

Cleanup and refactoring of Javascript to facilitate development of support for new features.

JIS-1671 / 5153539 - Maps Displayed After Natural Terminal Non Conversational Mode Activated
JIS-1694 - Implement page loading indicator

A page loading indicator has been added for XHTML clients. By default, a round animiated icon will (spinner) be displayed immediately after submitting a form to the server. The spinner will be displayed until the onLoad event of the subsequent page is invoked. The spinner can be delayed or disabled using the `<AppName>.ini [Xhtml] spinnerDelay` property setting.

```
[Xhtml]
spinnerDelay=<delayInMS>
```

Setting `spinnerDelay` to -1 will disable the spinner.

Setting `spinnerDelay` to a positive value will delay the spinner `<delayInMS>` milliseconds. For example setting `spinnerDelay` to 1000:

```
[Xhtml]
spinnerDelay=1000
```

will delay the spinner display 1 second and only display the spinner if the page takes longer than 1 second to load.

JIS-1682 / 5155150 - Upgrading to v9.2.1 - Javascript error
JIS-1683 - Prototype.js causing JavaScript error with IE8

A Javascript error introduced by the upgrade of `Prototype.js` from version 1.4 to 1.7 for JIS 9.2.1 has been fixed.

JIS-1693 / 5150037 - Date control issue
JIS-1711 - Date control fixes

The following date control issues have been fixed:

- Clicking on the date button does not open the calendar window
- When a date control on a tab control with more than one date control is updated, any other date control with an empty date setting is also updated with the same date.

JIS-1701 - Compress response to client

Reduce network traffic and response times to the client by compressing XHTML responses sent to the client when using a browser that supports GZip compression.

Compression is enabled by default when using a browser which supports GZip encoding. To disable GZip compression, add the following property setting:

```
[Http]
GZipEnabled=0
```

to the `jacadasv.ini` file.

When using the RedirectionProxy the old compression setting "UseGzip" has been removed, compression is enabled by default and depends on the new ini setting.

JIS-1727 / 5162398 - Application causing delay and loading on JITGUI
JIS-1744 - Fix emulator logic related to screen changes

A regression which affected the behavior of the `[emulator] AfterKeyboardUnlockWaitForScreenChangeBeforeIdentification <AppName>.ini` setting and could cause an application to hang on some screens has been fixed.

JIS-1754 / 5170343 - Delta - Need to inject <!DOCTYPE html> at the beginning of our page
JIS-1763 - Add support for the HTML5 DOCTYPE and remove XML declaration

The XHTML doctype tag is no longer included in pages generated for XHTML clients, instead, a `<!DOCTYPE HTML SYSTEM "about:legacy-compat">` declaration, as recommended by W3C, is added for all modern browsers. For IE9 and earlier versions no `DOCTYPE` declaration is included in the generated page, allowing the browser determine how to render it.

JIS-1760 / 5170962 - JIS application does not execute the prompting when prompt is issued from a table field
JIS-1761 - Prompt button inside a table does not work

A regression that caused the action value to be parsed from the post data of a table prompt incorrectly, causing the prompt button of the prompt inside a table (XHTML) not to execute the prompt method.

JIS-1778 / 5175923 - JIS 9.2.1 XHTML - View Host Screen
JIS-1783 - Remove code duplication between view host and printing buttons

If View Host Screen is enabled and printing is disabled, the View Host Screen button could overlay some data fields. The positioning of the View Host Screen button is no longer dependent on the Host Print button and will be placed on the bottom right of the browser window.

JIS-1780 / 5175421 - Radio Group issues
JIS-1782 - Fix radio button duplicate label on IE

Updates for Internet Explorer 10 support caused a regression that caused dynamic radio button labels not to override the static labels but to up below them.

JIS-1800 / 5180922 - Look of JacadaList is Corrupted in IE11
JIS-1835 - Adapt table and folded table to HTML5 rendering

Adapt table and folder table to HTML5 rendering. Add HTML5 tag when using popup support and remove unnecessary code.

JIS-1803 / 5182228 - JIS 9.2.1 getPostDate() API
JIS-1804 - Change the name attribute from txt to txtarea for text area in non-IE browsers

An incompatibility between HTML generated for Internet Explorer 9 and newer versions of IE or Netscape and Chrome browsers has been fixed. Text area components generated for Netscape / IE 11 browsers will now be prefixed with `"txtarea_"`.

JIS-1810 / 5183374 - Slow transactions when connected to secure host port
JIS-1814 - Specify TcpNoDelay when connecting to SSL server

The Java Socket option `TcpNoDelay` is now enabled when making an SSL connection to a host. This disables Nagle's algorithm for SSL host connections, potentially speeding up JIS server / host transactions.

JIS-1821 / 5184220 - Behavior In IE11 different concerning the display of the page when submitting it
JIS-1866 - Switching to IE11 compatibility mode automatically

It is now possible to switch IE11 into compatibility mode automatically. See “Enable Compatibility Mode in Internet Explorer 11 Automatically” on page 106 for a detailed description.

JIS-1826 - Fix JITGUI on IE11 with HTML5 Doctype

The HTML5 `<!DOCTYPE HTML>` declaration is now supported by the JITUGUI running on Internet Explorer 11 in native mode.

JIS-1848 / 5192870 - GEM - Missing XHTMLCSS Stylesheet Resource File - Causing "Expanding Tables"
JIS-1852 - Table CSS refresh problem

Caching of CSS stylesheet resources is now optional, see “Optional Caching of Table CSS Styesheet Resources” on page 106 for a detailed description.

JIS-1895 / 5211802 - Converting from java aplets to XHTML - Norwegian characters
JIS-1907 - Use UTF8 encoding when converting the static XML file to a DOM

Static XML text is now read using UTF-8 encoding to prevent corruption of the static text.

JIS-1912 / 5218237 - Clickjacking protection
JIS-1915 - Add support for the X-Frame-Options HTTP header to prevent clickjacking

The `XFrameOptions <AppName>.ini` setting has been added to mitigate clickjacking attacks. See “Clickjacking Protection” on page 105 for a detailed description.

JIS-1938 - Change click event behavior for date picker

When running an XHTML client session inside an HTA container. Selecting a date from the date picker window did not close the window. To fix this, the JavaScript function responsible for closing the window has been moved from the onClick event of the link to the link href.

Installation

JIS-1639 - Update code signing certificate

The code signing certificate has been updated.

JIS-1770 - Support for runtime installation Windows uninstall panel

Subapplications installed using the JIS runtime installer may now be uninstalled via the Windows Control Panel, using the uninstall dialog.

JIS-1904 - Update runtime installation to IzPack 5.0

The JIS runtime installer has been upgraded to IzPack 5.0. See “Runtime Installation Upgrade Instructions” on page 102 for instructions on upgrading existing installation projects.

JIS-1953 - 2016 code signing certificate

The code signing certificate has been updated.

Server

JIS-1607 / 5140355 - Delta - Excess Server Logs
JIS-1627 - Eliminate Session Dump when calling TotalExit

In XHTML, placing the `DoMethod TotalExit` after `MoveAccordingToHost` to terminate the session resulted in a session dump during the generation of the response. The session dump has been eliminated but an exception is still written to the server log.

JIS-1635 - Administrator runtime settings XHTML section contains empty checkbox

An empty checkbox added by mistake to the XHTML section of the Administrator runtime settings has been removed.

JIS-1651 / 5147560 - Server Disconnect on Text Entry screen
JIS-1652 - Problem sending data when a java.awt.TextArea is the focused control

Using a `java.awt.TextArea` from a Java extension and sending data to the server while the `textArea` component wass in focus, terminated the session with the following exception "java.lang.ClassCastException: java.awt.TextArea cannot be cast to cst.gwt.GUIControlInterface".

This has been fixed.

JIS-1669 - Upgrade to Jetty 9

The embedded Jetty HTTP server has been upgrade to version 9.4.2.

JIS-1678 - Use Servlet 3.0 AsyncContext in XHTML

Reduce resource requirements and improve scalability by using `AsyncContext` to manage the connection between a Java client and the Jetty HTTP server

JIS-1682 / 5155150 - Upgrading to v9.21.- Javascript error
JIS-1683 - Prototype.js causing JavaScript error with IE8

A Javascript error introduced by the upgrade of `Prototype.js` from version 1.4 to 1.7 for JIS 9.2.1 has been fixed .

JIS-1687 / 5155831 - Status of Java 8 certification
JIS-1691 - Java 8 certification

JIS Interface Server 9.2.2 is certified to run on Java 8.

JIS-1689 - Use AsyncContext in HTTP proxy servlet

Reduce resource requirements and improve scalability by using `AsyncContext` to manage the HTTP connection between a Java client and the JISProxyServlet.

JIS-1690 / 5155863 - Error popup during runtime - Couldn't create window
JIS-1692 - Dynamically allocate components per subapplication

The array used to hold Subapplication components is now allocated dynamically to reduce resource requirements and increase the limit on subapplication components from 1500 to 1800.

JIS-1704 / 1093683 - Password/hidden fields being logged.
JIS-1707 - Password information printed to server log

Several regressions that allowed password fields to be logged to the server log when the log level was set lower than 70 have been fixed.

JIS-1708 / 5158088 - JIS service status on JIS server
JIS-1740 - Do something about the missing -Xrs setting

When JIS is started as a Windows service registered using the `JBSToService.exe` utility, the startup code for the service, adds or updates the `JAVA_TOOL_OPTIONS` variable and sets its value to "`-Xrs`", this ensures the server starter process and the processes it spawns are using the `-Xrs` command line option by default and without additional configuration.

If the `-Xrs` option was already configured in `jacadasv.bat` or `jacadasv.ini` this does not cause any problem and the flag is still used.

JIS-1719 / 5161179 - Formats.ini file not working after ugrade of JIS software from 9.2 to 9.2.1
JIS-1726 - Fix dictionary upper case and separator symbols

A bug which caused the formatting dictionary to incorrectly parse the special characters used by the ini file to designate upper case letter and the equals separator as part of the text has been fixed.

JIS-1721 / 5162246 - POODLE Security Vulnerability Breaks SSLv3 Secure Browsing
JIS-1730 - Fix poodle security vulnerability

The SSLv3 and lesser protocols have been disabled to mediate the SSL POODLE attack vulnerability.

JIS-1727 / 5162398 - Application causing delay and loading on JITGUI
JIS-1744 - Fix emulator logic related to screen changes

A regression which affected the behavior of the `[emulator] AfterKeyboardUnlockWaitForScreenChangeBeforeIdentification <AppName>.ini` setting and could cause an application to hang on some screens has been fixed.

JIS-1737 - Java 8 support

Modifications required to support building and running on Java 8 while maintaining backward compatibility with Java 7.

JIS-1753 - HTTP to HTTPS redirection

The JIS HTTP port may be automatically redirected to the JIS HTTPS port. See "Automatic HTTP to HTTPS Redirection" on page 105 for a detailed description and example.

JIS-1802 / 5180721 - Many to One Screen issue
JIS-1808 - Many to One - do not collect messages from other dependents

Collection of messages from other dependents of a Many to One screen has been made optional and may now be configured using the `UpdateMessagesFromOtherSlaves` setting in the `jacadasv.ini` file. This behavior is enabled by default but may be disabled with the following `<AppName>.ini` setting:

```
[Program]
UpdateMessagesFromOtherSlaves=0
```

> **Note:** Customers who received a hot fix against previous releases of JIS for this problem will need to make the change described above.

JIS-1803 / 5182228 - JIS 9.2.1 getPostDate() API
JIS-1804 - Change the name attribute from txt to txtarea for text area in non-IE browsers

An incompatibility between HTML generated for Internet Explorer 9 and newer versions of IE or Netscape and Chrome browsers has been fixed. Text area components generated for Netscape / IE 11 browsers will now be prefixed with "`txtarea_`".

JIS-1811 / 5185330 - IP redirect
JIS-1815 - BindIPAddressForHTTPClient setting is broken

A regression which caused the `BindIPAddressForHTTPClient` setting in the `jacadasv.ini` file to be ignored has been fixed.

JIS-1810 / 5183374 - Slow transactions when connected to secure host port
JIS-1814 - Specify TcpNoDelay when connecting to SSL server

The Java Socket option `TcpNoDelay` is now enabled when making an SSL connection to a host. This disables Nagle's algorithm for SSL host connections, potentially speeding up JIS server / host transactions.

JIS-1829 / 5189161 - Jetty URL case senitivity and ETag support
JIS-1838 - Enable Etag response header to improve caching

JIS now supports Etags or entity tags. Etags are a mechanism provided by the HTTP protocol for web cache validation, which allows a client to make conditional requests, potentially allowing web caches to be more efficient and reducing bandwidth. Etags are enabled by default but can be disabled with the following `jacadasv.ini` property setting:

```
[Http]
EnableEtags=0
```

JIS-1846 / 5192404 - Package process for JIS - target root directory coming out in all caps
JIS-1847 - Application Server deployment - fix case of RootDir parameter

The `RootDir` parameter in the `web.xml` generated for an application server deployment is no longer forced to be uppper case.

JIS-1863 / 1107543 - Issue with scroll bars on tables
JIS-1870 - Subapplication specific screen system trigger methods are not invoked

A regression which caused subapplication specific trigger methods for the screen receiver, such as `PageUp`, `PageDown`, `GetToBottomOfList`, `GetToTopOfList`, etc. not to be invoked has been fixed.

JIS-1888 / 5210202 - Unable to start application using an obfuscated JettyKeyStore password
JIS-1893 - SSL socket cannot read obfuscated keystore password

Obfuscated Java keystore passwords are now supported for SSL socket communication between Java clients and the JIS server.

JIS-1895 / 5211802 - Converting from java aplets to XHTML - norwegian characters
JIS-1907 - Use UTF8 encoding when converting the static XML file to a DOM

Static XML text is now read using UTF-8 encoding to prevent corruption of the static text.

JIS-1912 / 5218237 - Clickjacking protection
JIS-1915 - Add support for the X-Frame-Options HTTP header to prevent clickjacking

The `XFrameOptions <AppName>.ini` setting has been added to mitigate clickjacking attacks. See “Clickjacking Protection” on page 105 for a detailed description.

JIS-1966 / 5244901 - Printing Issue
JIS-1968 - Improve printing of image buttons

Most of the GIF button images provided now print with a transparent background.

JIS-1973 / 5248661 - Multiple JIS clients continually freeze
JIS-1974 - Reduce PS items log verbosity

Verbose logging of presentation space elements is now restricted to log level 100.

JIS-1977 / 5251062 - Jetty web server configuration question
JIS-1978 - Disbable unused HTTP methods and protect server identity

The `TRACE`, `OPTIONS` and `HEAD` HTTP methods have been disabled in the Jetty configuration for security purposes.

JIS-1993 - Load the JIS administrator as a Java Web Start application

The JIS Adminstrator utility may be loaded as a Java Web Start application. See "Java Web Start Support" on page 104 for a detailed description.

JIS-1994 - Wize runtime installation - JIS server failed to start

The JNLP resources are now included in the Wise runtime installation so the server can be started.

Limitations of JIS 9.2.2

The following are known limitations in JIS 9.2.2:

- JIS-1670 - Cannot connect to emulator to capture screens
  JIS-1679 - Java based screen capture tool
- JIS-1647 - Software Flow Control
  JIS-1700 - Typing data using a driver license scanner
- JIS-1697 - Specific Customizations
  JIS-1713 - Remember default runtime installation folder
- JIS-1738 - Unsigned Java client cannot connect to server using SSL

- JIS-1793 - Project specific configuration for page loading indicator
- JIS-1849 - JIS server performance issue
  JIS-1851 - Increase the number of Jetty threads automatically
- JIS-1830 - DB Bug - Screen
  JIS-1877 - Delete subapplication do not delete the screen image from the database
- JIS-2000 - Administrator - Runtime Configuration Tab -Can't open the "Category" combo

JIS-1670 - Cannot connect to emulator to capture screens
JIS-1679 - Java based screen capture tool

The JIS development environment does not support secure host connections. The runtime environment does support secure host connections providing a possible workaround by going to the menu item: Application / Emulator / Save Host Screen Image...

JIS-1647 - Software Flow Control
JIS-1700 - Typing data using a driver license scanner

JIS does not support the use of scanners for data input to applications.

JIS-1697 - Specific Customizations
JIS-1713 - Remember default runtime installation folder

The ACE interactive development tool does not remember user selections for runtime installation path.

JIS-1738 - Unsigned Java client cannot connect to server using SSL

When creating a socket connection to the server, the Java applet first requests the file `crossdomain.xml` which grants the client permission to handle data across multiple domains. The server `crossdomain.xml` file is auto-generated by the server when it receives a `GET /crossdomain.xml` request from the client and the socket connection is allowed due to the cross domain permissions. This work around does not work for SSL connections.

JIS-1793 - Project specific configuration for page loading indicator

The page loading indicator introduced for this release is not currently customizable or configurable per project. See “JIS-1671 / 5153539 - Maps Displayed After Natural Terminal Non Conversational Mode Activated JIS-1694 - Implement page loading indicator” on page 114 for details.

JIS-1849 - JIS server performance issue
JIS-1851 - Increase the number of Jetty threads automatically

It is not currently possilbe to dynamically increase the number of threads Jetty allocates for connection handling. As workaround, the Jetty `maxThreads` setting should be set manually to the expected number of concurrent connections + 20% but ideally no more than 500 depending on resource consumption.

JIS-1830 - DB Bug - Screen
JIS-1877 - Delete subapplication do not delete the screen image from the database

Screen images are not deleted from the database when a subapplication is deleted. The workaround requires manual editing of the JIS application database. Please consult Software GmbH Global Support for assistance.

JIS-2000 - Administrator - Runtime Configuration Tab -Can't open the "Category" combo

The Category drop down dialog on the Runtime Configuration tab in the JIS Administrator utility does not work correctly when Administrator is started as a Java Web Start application. The drop down dialog can be navigated with the arrow keys after being selected via the mouse or tab controls. This limitation only applies to the Adminstrator utility when started as Java Web Start application.

JIS Interface Server 9.2.1 Release Notes

Installation & Upgrade Information

Supported Platforms

JIS Interface Server 9.2.1 is supported on the following platforms:

- Windows Server 2003 Standard and Enterprise Edition (32-bit)
- Windows Server 2008 Standard and Enterprise (32-bit)
- Windows Server 2008 Standard and Enterprise (64-bit)
- Windows Server 2012 Standard and Enterprise (64-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (32-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (64-bit)
- Windows 8.1 Professional, Ultimate and Enterprise Edition (64-bit)
- Solaris SPARC 10 (64-bit)
- Solaris SPARC 11 (64-bit)
- AIX 6.1 Power (64-bit)
- AIX 7.1 Power (64-bit)
- Red Hat Enterprise Linux 6 for x86 (64-bit)

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for Java versions, browser versions and application server versions supported by their respective vendors. Being so, when a vendor no longer supports a Java version, browser version or application server version, Software GmbH will discontinue supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows XP Professional SP3
- Windows 7
- Windows 8.1

Clients

The Java Client has been tested on the following operating systems, browser and Java versions:

- Windows 7
- Windows 8.1

Table 7 - 23 : JIS 9.2.1 Java Client Supported Browser and Java Versions

Browser	JRE
IE 8, 11	Oracle 1.7.0
Firefox	Oracle 1.7.0
Chrome	Oracle 1.7.0

When using Java 1.7.0_51 and higher all the components of the Java Applet have to be packaged into Jar files and digitally signed. This process is explained in the "Runtime Installation" section. However, during development you may configure the Java JRE so that it is not necessary to package and sign all class files and resources as explained below.

When launching the Java client in the development environment for the first time you are prompted to download and install the Java JRE from the Java.com web site.

After installing the Java JRE 1.7.0_51 or higher and launching the client Applet you'll receive the following error message:

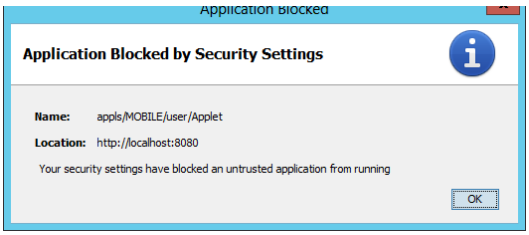

Figure 1. JIS 9.2.1 Application Blocked by Security Settings

The reason for this message is that the class files and resources generated by the "Generate Runtime" process are not digitally signed.

In order to eliminate this message during development, reduce the Java Plugin security level to "Medium" or add the server address and port to the "Exception site list" as shown below. These configurations are accessible from the Java Control Panel "Security" tab.

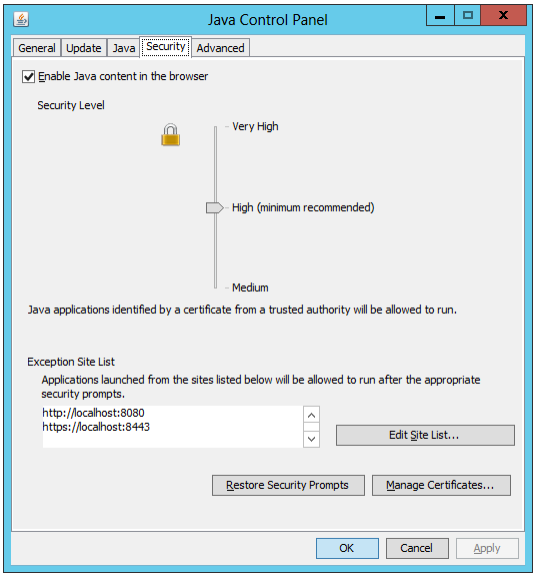

Figure 2. JIS 9.2.1 Control Panel Security Tab

Now restart your browser and launch the Java client Applet again. You'll now receive the following confirmation dialog:

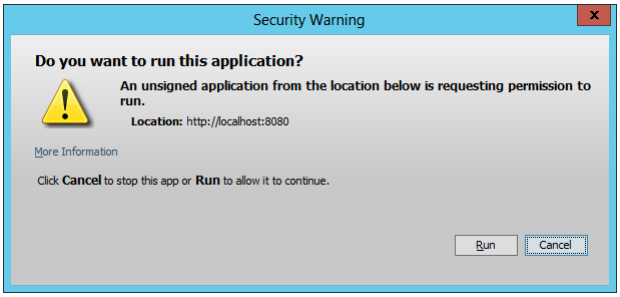

Figure 3. JIS 9.2.1 Java Security Warning Dialog Box

Click the <Run> button to launch the Applet.

The XHTML client has been tested with the following operating system and browser versions:

Table 7 - 24 : JIS 9.2.1 XHTML Client Supported Operating Systems and Browsers

Operating System	Browser
Windows	IE 8, IE 11
Windows	Firefox
Windows	Chrome
Mac OS/X	Safari
iPad iOS	Safari Mobile

Standalone Server

The JIS standalone server has been tested in the following environments:

Table 7 - 25 : JIS 9.2.1 Standalone Server Supported Operating Systems

Operating Syste	Java Version
Windows 2003	Oracle 1.7.0
Windows 2008	Oracle 17.0

Table 7 - 25 : JIS 9.2.1 Standalone Server Supported Operating Systems

Operating Syste	Java Version
Solaris 10 & 11	Oracale 1.7.0
AIX 6.1 & 7.1	IBM 1.7.0
Red Hat Linux AS6	Oracle 1.7.0

Application Server Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 7 - 26 : JIS 9.2.1 XHTML Client Supported Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WebSphere 8.5.5	IBM JRE 1.7	Windows 2003
Tomcat 8	Oracle JRE 1.7	Windows 2008

OS400 Components

The DDS compiler has been tested on the following operating systems:

- OS400 V6R1
- OS400 V7R1

Upgrade Instructions

The structure of the launcher page for the Java client <AppName>.html has changed. Sites which customized the launcher page need to adjust their customized page to the new structure of the Applet tag.

Make sure to incorporate the following changes:

1. Modify the archive attribute to the following value:
   `archive="cst/clbase.jar,${APPL_ARCHIVE}"`
2. Add the following Applet parameter to support pack200 archive format:
   `<param name="java_arguments" value="-Djnlp.packEnabled=true"/>`

Retirement of Product Components

As of release 9.2.1 the following product components are retired:

- Wise Runtime Installation – replaced by the IzPack runtime installation. Wise installer functionality is still part of the product in order to allow for smooth migration.
- Session Pre-loading – this feature was introduced in V9.0 as the first step towards full-fledged session pooling implementation. However, session pooling is no longer in the product roadmap and the feature in its current state does not provide significant benefit.
- Online help for the Administrator utility – removed, since the JavaHelp 3rd party component on which it relies has not been updated by its vendor.

As of release 9.2 the following product components were retired:

- Innovator – removed as part of migration to 64-bit
- Jacada Connects – sites still using Jacada Connects should continue to use their current version and discuss their project roadmap with Software GmbH's product management
- Deploying the standalone server to AS/400 – sites should switch to deploying the server to Windows or Unix platforms
- Application Clustering – sites using this feature should switch to deploying the clustered applications as separate applications
- Jacada Common Installation for application servers – no longer required since the application ear file now includes all product components
- Standalone Java Client Proxy – replaced by internal server functionality

Note: Extended maintenance for previous versions can be requested from Software GmbH.

New Features in Version 9.2.1

The following new features have been added for JIS 9.2.1

- Runtime Installation
- Enhanced Java Client Packaging Procedure

- Parameter Substitution in HTML Launcher Page
- Pass Through Gateway
- View Host Screen in XHTML
- Simplify Redirection Proxy Configuration
- Secure Single Signon Implementation using Redirection Proxy
- Proxy Servlet to Server Direct Communication
- Support Underline and Strike-through Font Attributes
- Solaris KSSL
- Jetty HTTP Thread-pool Monitoring
- Server Business Logic
- IE10 & IE11
- Server Remote Management
- Remote Debugging
- BIN2XML Utility
- Mobile Demo Web Site
- Logging Improvements

Runtime Installation

The discontinued “Wise Installation Studio” product which used to be the JIS recommended tool for creating the runtime installation is no longer supported by its vendor.

We have now introduced a new runtime installation procedure based on the IzPack product.

IzPack is a widely used tool for packaging applications on the Java platform. IzPack is open source (Apache license) and free. IzPack is now part of the JIS product installation so there is no need to purchase a 3rd party tool.

By using IzPack, we provide the same level of functionality and configuration of the Wise runtime installation but at no additional cost and with simpler installation procedure.

The IzPack installation supports the following existing Wise functionality:

- Graphical user interface
- Full installation and library update
- Windows and Unix installation
- Configurable installation scripts which customers can re-brand
- Creation of menu shortcuts on Windows platforms
- Creation of Windows executable

In addition IzPack also supports:

- Headless and X-Windows based installation on Unix platforms
- Readme and license agreement dialogs
- Packaging and signing the Java client Jar files as part of packaging the runtime installation

Limitations

- The runtime installation wizard will no longer generate a text file with the list of files to install
- Installing the runtime on AS/400 is no longer supported (using an AS/400 host is still supported)

In order to simplify migration, the Wise related functionality can still be used but is no longer maintained.

Creating the IzPack runtime installation

From ACE, invoke the “Create Runtime Installation” wizard.

The “Select Installer Type” step of the “Create Runtime Installation” wizard has a new option “Create IzPack installation”.

Select this option and click “Next” to create an IzPack based runtime installation.

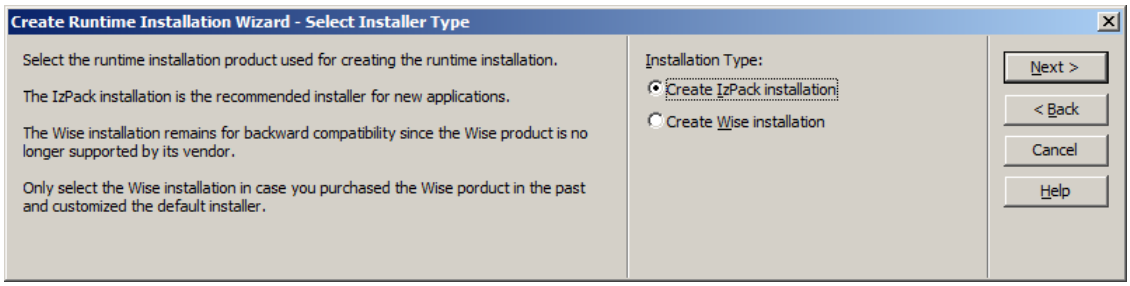

Figure 4. JIS 9.2.1 Create Runtime Installation Wizard - Select Installer Type Dialog Box

The rest of the wizard steps are similar to the Wise installation steps.

Finish the “Create Runtime Installation” wizard to package the runtime installation. Packaging related messages are logged to the console window.

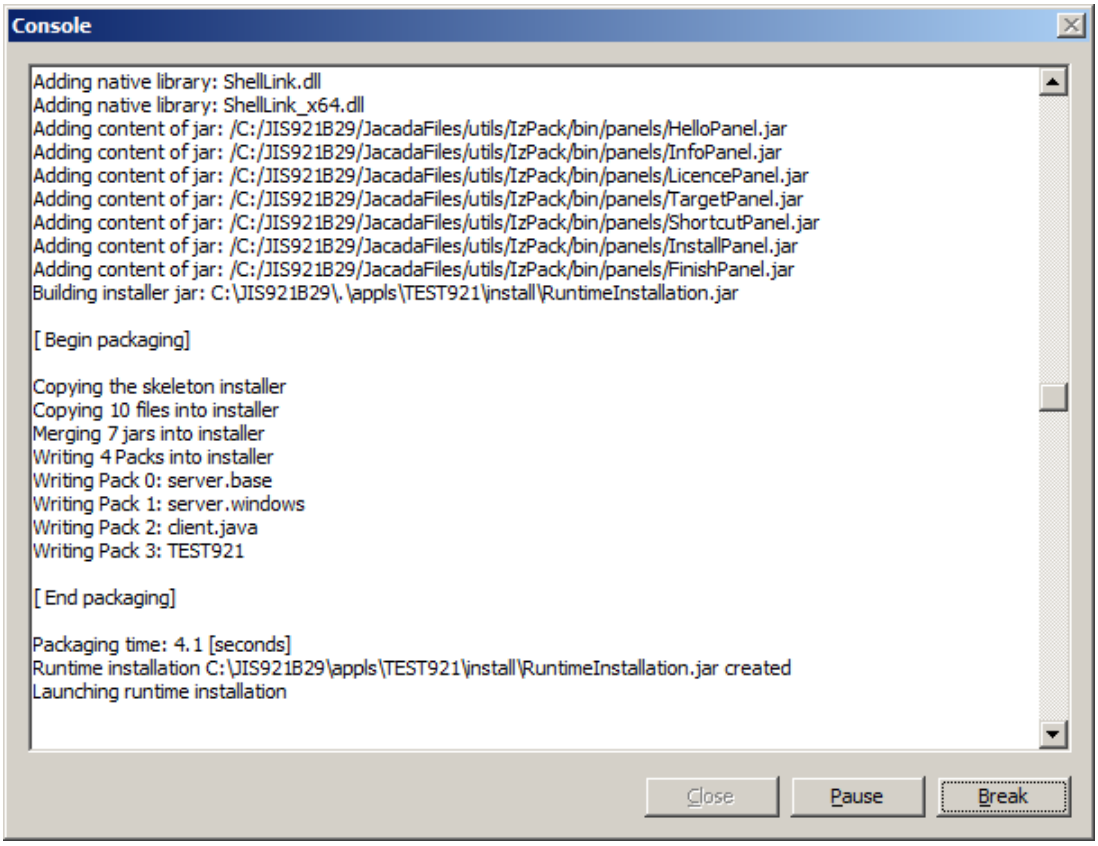

Figure 5. JIS 9.2.1 Create Runtime Installer Console Window

Once the packaging is done the installer window is launched automatically.

If you do not wish to proceed with the runtime installation on the local workstation click “Quit”. The installation files are created in the following path: `<JISRoot>\appls\<AppName>\install`

The runtime installation is now ready to deploy to a remote machine.

To continue the installation on the local workstation:

1 Click “Next”.

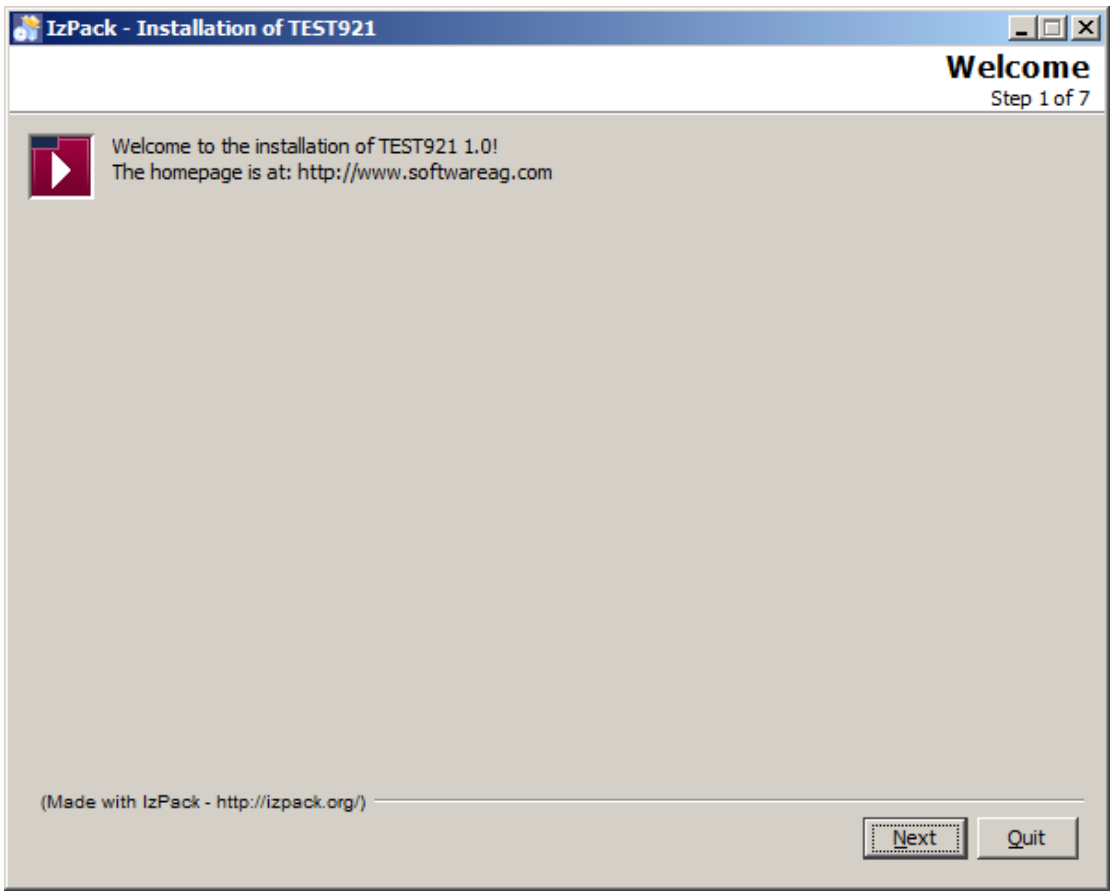

Figure 6. JIS 9.2.1 IzPack Welcome Dialog Box

2 Read the readme text and click “Next”.
3 Accept the license agreement and click “Next”.
4 Select the installation path and click “Next”.
5 On Windows platforms, setup the windows shortcuts and click “Next”.
6 Wait for the installation to finish, click “Next”.
7 Installation finished, click “Done”.

Installing the runtime on a remote machine:

1 Make sure a supported “Java JDK” or “Java Server JRE” is installed on the target machine, and that the Java command is in your machine path. You may install the runtime using a “Java JRE”, however the installed server cannot run using a “Java JRE” since it requires a server grade Java installation.
2 Copy the RuntimeInstallation.jar file located in the `<JISRoot>\appls\<AppName>\install` folder to the target machine. For Unix platforms, use binary ftp when transferring the file.
3 On the remote machine, open a command window and type:
```
java -jar RuntimeInstallation.jar
```

4. On a Windows machine, the installer dialog is launched.
5. On a Unix machine, if the X-Windows environment is configured the graphical runtime installation dialog is launched. Otherwise, a command line installation is invoked with similar installation steps.

Creating an installation kit

To package the runtime installation into a distributable deployment package, follow these steps (Windows only):

- Copy the files RuntimeInstallation.jar, RuntimeInstallation.exe to the root folder of your installation kit.
- Make sure that Java 7 JDK is installed on the target machine. This can be configured as part of your deployment procedure but it's not part of the runtime installation.
- Launch the installer by launching RuntimeInstallation.exe from the installation kit.

Installer customization:

The installation metadata files in folder `<JISRoot>\appls\<AppName>\install` can be customized to modify the installer look & feel.

The following files can be customized:

- `install.xml` – standard IzPack installation script – you may modify this file to customize your installation.
- `shortcutSpec.xml` - for Windows installations, defines the shortcut menus generated by the installation.

For further instructions about the structure of these files, consult the IzPack documentation at:

`<JISRoot>\JacadaFiles\utils\IzPack\doc\izpack\pdf\manual.pdf`

Add your application specific license agreement to `license.txt`.

Add your application specific readme messages to `readme.txt`.

Once you are done with your customizations. Use the "Create Runtime Installation" wizard to generate a new installer which will automatically incorporate your changes.

File override policy

When installing a new runtime installation on top of an existing runtime installation the following update policy takes place:

The following configuration files are not overwritten by the runtime installation:

- Server start-up script (`jacadasv.bat` on Windows, `jacadasv` on Unix, these files are located in the bin folder)
- Server ini file (`jacadasv.ini` in the classes folder)
- Application ini files (`<AppName>.ini` and `formats.ini` in the `classes\appls\<AppName>\server\resources` folder)
- Application HTML files (`<Appl>.html`, `<Appl>-xhtml.html` in the root folder)

The rest of the installed files are always overwritten.

Uninstalling a runtime installation

Use the following procedure to uninstall a runtime installation

- Make sure the server and JIS administrator are not running
- Open a command window and navigate to the root folder of the runtime installation
- Run the following command:
  `java -jar Uninstaller\uninstaller.jar`
- The Uninstaller dialog opens

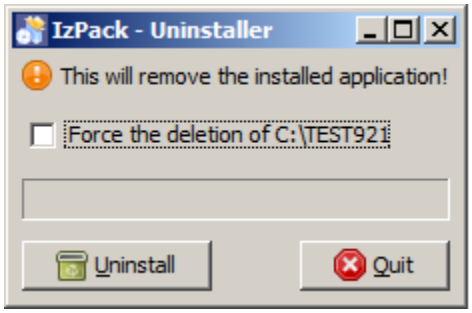

Figure 7. JIS 9.2.1 IzPack Uninstaller Dialog Box

- Check the “Force the deletion of …” checkbox and click the <Uninstall> button to remove the runtime installation

Enhanced Java Client Packaging Procedure

With the introduction of the Java 1.7.0 update 51 JRE, unsigned Java class files, Jar files and other resource files, are blocked by default. Therefore unsigned product Jar files are no longer usable and auto generated application class files must be packaged into Jar files and digitally signed by the customer before deployment to production.

In addition Java now supports the pack200 compression algorithm which compresses Jar files to up to 30% of their original size.

The following changes were implemented in the product to adapt the Java client deployment procedure to take advantage of these changes:

1. The old `clbase.jar`, `clbase-signed.jar` and `clfull-signed.jar` were replaced with a single client Jar file named `clbase.jar`. This file is equivalent to the old `clfull-signed.jar` and is digitally signed by Software GmbH. In addition the product now includes the `clbase.jar.pack.gz` file which is a compressed pack200 package generated from `clbase.jar`.
2. Similarly the administrator utility jar file `jam.jar` is now digitally signed and a pack200 compressed package `jam.jar.pack.gz` is distributed.
3. The product now generates only a single `<AppName>.html` Java client launcher file pointing to the `clbase.jar` file.
4. The archive attribute of the Applet tag now contains the `${APPL_ARCHIVE}` token. During the packaging of the runtime installation, this token is replaced with a list of Jar files generated by the new IzPack runtime installation. The runtime installation generates and optionally signs a Jar file per application library.

Running the Java client in the development environment

The generate runtime process generates Java source code based on the designed subapplications and compiles it into class files. Since these class files are unsigned, JRE 1.7.0_51 will not load these classes when using its default configuration.

To configure the Java JRE to load unsigned classes during development, set the Java Control Panel security level to “Medium” or add the local server address and port to the exception list.

See the “Java Control Panel” security tab for more information.

Packaging application resources into Jar files

The Makejar utility, now supports packaging application specific class files and resources into signed Jar files, and packaging these Jar files using pack200 into highly compressed archives.

The same mechanism used by the MakeJar utility is also used by the IzPack installer to package application resource when generating the runtime installation.

The MakeJar utility is invoked by executing the makejar.bat under the `<JISRoot>\JacadaFiles\utils\makejar` folder.

By default, it will package an unsigned Jar file with the client application resources.

Optionally, provided the necessary information, it will also sign the Jar file using a digital signature in the form of a Java keystore provided by the customer.

In order to digitally sign the generated Jar files, Makejar relies on the following `jacadasv.ini` parameters under the `[makejar]` section:

- `jarsigner` - a flag to activate the Jar signing utility. Values: 0,1 [default: 0]
- `Keystore` - fully qualified path to a standard Java Keystore file containing the code signing certificate
- `storepass` - the keystore password, this password can be obfuscated
- `storetype` - keystore type as specified by the Java keytool utility
- `tsa` - fully qualified url of the time stamping service (optional). Time stamping your Jar file at the time of signing ensures that the signature will never expire.
- `alias` - the alias of the code signing certificate

Example11. JIS 9.2.1 Makejar Jar Signing Properties:

```
[MakeJar]
jarsigner=1
Keystore=c:\sign\mykeystore.pfx
storepass=OBF:1i8m1irz1ge513en1ght1iun1i9d
storetype=PKCS12
tsa=https://timestamp.geotrust.com/tsa/
alias={972e90db-cbbd-4bfc-80af-fb1a6f39292a}
```

For more information about the `keystore`, `storepass`, `storetype`, `tsa` and `alias` parameters consult the documentation of the Java JarSigner utility.

For more information about creating obfuscated password for use as the storepass parameter see the section “Secure Login to JISAdminServlet” on page 260.

Parameter Substitution in HTML Launcher Page

JIS client sessions are launched by pointing the client browser to one of the following HTML pages:

`<AppName>.html` – launches a Java client session.

`<AppName>-xhtml.html` – launches an XHTML client session.

It is now possible to add, dynamic, server based data to these static HTML pages.

Inside the HTML page used for launching the JIS session specify substitution parameters in the following format:

```
${<context>:<property>}
```

Where `<property>` is the case sensitive name of a property or parameter and `<context>` is one of the following literal strings explained below: prop, system, header, param, cookie

Context values explanation:

Table 7 - 27 : JIS 9.2.1 Client HTML Context Values

Name	Description

Prop	Name of a property in a standard Java properties file loaded by the server.
	
	The name of the property file in which this property is defined is determined by the following `jacadasv.ini` setting:
	
	`[Http]`
	`PropertiesFilePath= <file name>`

System	Name of a Java system property obtained using the Java API `System.getProperty()` invoked by the server
Header	Name of an HTTP header sent by the client when requesting the launcher page. The HTTP header value replaces the token in the launcher HTML
Param	Name of GET or POST parameter sent by the client when requesting the launcher page. The HTTP header value replaces the token in the launcher HTML
Cookie	Name of a cookie sent by the client when loading the launcher page from the server

To define the list of launcher HTML pages affected by this feature, in the `jacadasv.ini [Http]` section set the value of the setting `ParameterizedTargets` to a semicolon separated list of resources on which parameter substitution should be performed. Only these resources are affected by parameter substitution.

Example 12. JIS 9.2.1 Setting the Value of the DebugLevel Applet Parameter

Setting the value of the `DebugLevel` Applet parameter inside `<AppName>.html` based on the value of the property `debug.level` in the Java property file `/home/myuser/server.properties`

Set the following parameter in `<AppName>.html` files:

```
<PARAM name = "DebugLevel" value = "${prop:debug.level}">
```

Add the following settings to `Jacadasv.ini`

```
[Http]
ParameterizedTargets=/MYAPP.html
PropertiesFilePath=/home/myuser/server.properties
```

Create the file `/home/myuser/server.properties` and add the following line:

```
debug.level=70
```

During runtime when the browser loads an HTML page from the server, the server will replace the parameters with their substitution values and send the response page back to the client.

The sequence of events to launch the Java client would be as follows:

1 Browser requests to load an HTMLpage

2 Server checks if the page appears in the list of parameterized targets

3 Server loads the HTML

4 Server performs parameter substitution based on the parameter context defined in the page

15 Server writes the modified page back to the response

6 Client uses the response page to load the Applet

Pass Through Gateway

> **Note:** This is an advanced feature which requires good understanding of web technologies.

In order to closely integrate a JIS session with other web applications it is sometimes desirable to use the connection to the JIS session and the other web application over the same server address and port. This can be accomplished by using the JIS server as a gateway to an external web application.

To define the gateway mapping add the *`[GatewayMappings]`* section to `jacadasv.ini` and define a list of mapping settings in the following form:

```
[mapping.logical.name]=[resource.path],[gateway.to.url]
```

Where:

`[mapping.logical.name]` is an arbitrary string representing a descriptive unique identifier of the mapping.

`[resource.path]` is the prefix added to the JIS web site address to identify a mapped resource.

`[gateway.to.url]` is the URL address from which all requests to the `[resource.path]` should be served.

Example 13. JIS 9.2.1 Setting JIS GatewayMappings Properties

To enable JIS sessions to access information from a web site such as http://www.w3c.com over the same server address and port. Add the following mappings:

```
[GatewayMappings]
w3c.main=/w3c,http://www.w3.org/
w3c.css=/2008,http://www.w3.org/2008
w3c.2009=/2009,http://www.w3.org/2009
w3c.standards=/standards,http://www.w3.org/standards w3c.consurtium=/
Consortium,http://www.w3.org/Consortium/ w3c.participate=/
participate,http://www.w3.org/participate/
```

Then assuming that the JIS server listens on localhost:8080 you can access the w3c web site using the following URL: http://localhost:8080/w3c.

Under the hood, the server is using an internal proxy servlet which accepts client requests to the context path URL (resource.path) and forwards them to the target web site (gateway.to.url). The server then reads the response from the target web

site and writes it back to the client browser. In case the target web site contains relative links starting with a '/' the context path of these links should be added to `[GatewayMappings]` section, see for example the w3c.css mapping in the example above which enables the w3c web site to locate its CSS files using a relative path /2008 which passes through the JIS gateway to url http://www.w3.org/2008

Limitations

JIS cannot serve as a gateway to any site, sites that rely on advanced web technologies or advanced encryption technologies might be incompatible with the gateway functionality.

View Host Screen in XHTML

When using the Java client, users were always able to view, print and save the emulator screen. We have now introduced this functionality into the XHTML client. To enable XHTML users to view the host screen add the following `<AppName>.ini` setting:

```
[Xhtml]
ViewHostScreenEnabled=1
```

When setting `ViewHostScreenEnabled=1` a new "View Host Screen ..." button is added to each sub-application page at the bottom right of the web page. Clicking the new "View Host Screen ..." button displays the "View Host Screen" page in a new browser tab. The "View Host Screen" page presents a read only snapshot of the current host screen.

The left pane of the "View Host Screen" page contains the following widgets:

- "Show Line Numbers" checkbox - toggles line number display on the host screen.
- "Show Attributes" checkbox - toggles field attribute display. Input attributes are marked with the '`@`' symbol, output attributes are marked with the '`&`' symbol.
- "Refresh" button - updates the "View Host Screen" page to present the current host screen and reflect changes to the "Show Line Numbers" and "Show Attributes" checkboxes.
- "Save" button - saves a screen image file on the server machine. The screen image is saved as a panel file in the logs folder of the server. The "View Host Screen" page displays the path to the saved file on the server machine.
- "Download" button - downloads a screen image file, in panel format, to the client workstation.
- "Print" button - prints a textual representation of the screen image to the workstation printer.

> **Note:** Screen image files are saved as .pnl files. This panel file type is used for creating screen images from a screen capture in the design environment.
> During development the "Save" and "Download" actions differ only in the path where the screen image file is saved. However in production configuration, using the "Save" button for saving the file on the server side provides the system administrator additional information about the subapplication name, session number and process id of the screen image from which the file was saved.

Simplify Redirection Proxy Configuration

The function of the XHTML RedirectionProxy servlet is to expose the multiple processes used by the server using a single HTTP/S port to an external client. Historically, this servlet was designed to be deployed separately from the server itself into an external servlet engine such as Tomcat. However in practice the servlet is always running as part of the integrator process (1.0) of the JIS server itself. Therefore the independent proxyConfiguration.xml file is no longer necessary and the servlet configuration can either be automatically set by the server or configured using the standard `jacadasv.ini` file.

The following optional settings were added to the `jacadasv.ini` `[RedirectionProxy]` section:

- `IPAddress` – IP address of the server. This setting is now determined automatically in most configurations. Note: never use the value "`localhost`" or `127.0.0.1` for this setting as this will cause sessions to switch servers when using multiple servers behind a load balancer.
- `Port` – port of the server root process. Set it to the HTTP port used by the server root process. This setting is determined automatically in most configurations.
- `UseGzip` - set to "`1`" or "`yes`" to compress the response to the client using GZip. Default "`no`".
- `CloseStreamToClient` - set to "`0`" or "`no`" to prevent closing the client response stream by the server (this helps working through a reverse proxy). Default "`yes`".
- `BypassAfterRedirection` - set to "`1`" or "`yes`" to configure the client to redirect directly to the worker process bypassing the proxy. Default "`no`".

The file `proxyConfiguration.xml` has been removed from the product installation and runtime installation.

Backward compatibility

As long as your installation contains the (now obsolete) `proxyConfiguration.xml` file and the old `ProxyConfigurationFile jacadasv.ini` setting to locate it, the settings in this file are still in use. However we recommend moving the existing `proxyConfiguration.xml` settings to the `jacadasv.ini [RedirectionProxy]` section and deleting the `proxyConfiguration.xml` file.

The sequence of looking up the parameter values is:

1. `proxyConfiguration.xml`
2. `[RedirectionProxy]` section in `jacadasv.ini`

The actual settings being used in runtime can be monitored by looking up the "RedirectionProxy configuration" message in `debug_1.0.log` at debug level 50 or higher.

If specific settings do not exist in both of these files then the following defaults are used:

1. Server name set to the IP address of this server.
2. Port number set to the port of the root process.
3. All other parameters are set to their default values.

Following this enhancement the only `jacadasv.ini` settings required for setting up the integrator process (1.0) as a redirection proxy are:

```
[GeneralParameters]
MaxProcesses=<Value larger than 2>

[M1.Integrator.Sessions]
MaxProcessSessions=0
```

Having these two settings in your `jacadasv.ini` will enable the RedirectionProxy on the HTTP/S port used by the integrator process.

Secure Single Signon Implementation using Redirection Proxy

JIS XHTML now supports a secure mechanism for passing user credentials from the client browser to the mainframe.

Pre-requisites

- Browser communicates with the server using HTTPS using the RedirectionProxy servlet.
- A web page or other mechanism is defined for posting user credentials to the server using an HTTP POST request. Example of credentials entry page:

```
<html>
   <form action="XHTML?JacadaApplicationName=MYAPP" method="post">
      <input name="user"/>
      <input name="password"/>
      <input type="submit">
   </form>
</html>
```

The signon process works as follows:

- Initial connection from the browser to the redirection proxy is implemented using a POST request.
- Redirection proxy connects to the root process and then redirects to the worker process.
- Data posted by the browser
- to the redirection proxy is now posted again to the worker process – this is the new behavior which did not exist before.
- Worker process maps the posted data to shared user variables.
- Once the connection to the mainframe is established the data posted by the browser is typed into the mainframe signon screen to perform the signon operation.

To improve performance, it is now possible to configure the redirection proxy to bypass itself after the initial signon is performed. This way the redirection proxy is used only for the initial connection and is then bypassed by the client which connects directly to the worker process for the lifetime of the session.

To configure this option add the following `jacadasv.ini` setting:

```
[RedirectionProxy]
BypassAfterRedirection=1
```

Proxy Servlet to Server Direct Communication

The JISProxyServlet which is used for HTTP tunnelling of the Java client communication to the server is now part of the server itself. Each server process, can also serve as a JISProxyServlet.

In previous versions, even though both the proxy session and server session were running in the same server process, the communication between them was managed using socket communication.

In our recommended configuration this works well, the proxy servlet is running as part of the integrator process (1.0) and communicates with the other server processes using sockets with localhost address.

However, when scaling the server to over 1000 concurrent sessions this architecture has the following limitations:

1. Each proxy session creates two sockets to the worker process and spawns two threads to listen on these sockets. This creates a large number of threads which may exhaust the resources of the Java process or the server machine itself.
2. When the server is under heavy load, garbage collection cycles start taking significant time. If one of the processes is performing a lengthy garbage collection cycle and the other process is waiting for information, the result is EOFException and SocketException errors causing sessions to disconnect.

To improve the scalability of this architecture, we have now introduced a new behavior triggered by the following `jacadasv.ini` setting:

```
[JISProxyServlet]
IsOverrideConnectPort=1
```

When using this setting the proxy servlet creates the server session in the same process on which it is running and communicates with the server session using direct calls instead of using socket communication. This enhancement, improves performance and prevents disconnects caused by garbage collection since all communication is performed by the same server process. In addition, the number of threads per session is reduced. This configuration can be used in two scenarios:

1. Single process server.
2. Multiple process server behind a load balancer where the load balancer is responsible for distributing the sessions between the worker processes.

Limitations

In multiple processes configuration, connecting a client to the root process still uses socket communication. Make sure clients connect directly to the worker process 1.1 – 1.n

Support Underline and Strike-through Font Attributes

The “Strikeout” and “Underline” properties in the ACE “Font and Color” dialog are now supported by the Java client. Once defined in ACE, these properties will automatically affect the font decorations used by the Java client.

In XHTML the “Strikeout” and “Underline” properties cannot be used together for the same component. In case both are specified, only the “Underline” property is used.

Solaris KSSL

To support encrypted communication between the browser and the server, the server is using Java Secure Socket Extension by default.

To improve performance when deploying the server to a Solaris operating system, use the KSSL operating system service which relies on the operating system encryption kernel.

KSSL is now supported by the JIS server.

Jetty HTTP Thread-pool Monitoring

The following KPIs are now displayed in the process properties panel of the JIS administrator utility:

- `Stat.1.Http.Min.Thread` - minimum number of HTTP threads
- `Stat.2.Http.Max.Thread` - maximum number of HTTP threads
- `Stat.3.Http.Idle.Thread` - number of idle HTTP threads waiting in the pool
- `Stat.4.Http.Is.Low.On.Thread` – this value is set to true when the server is unable to allocate HTTP threads for new HTTP requests – this is a critical condition which severely affects server response times.

By default the minimum thread pool size is set to 10 threads and the maximum is set to 200. In certain configurations, this number should be increased in the http.xml and proxyHttp.xml configuration files.

Use the new KPIs to monitor the HTTP thread pool behavior.

Server Business Logic

In order to further improve the server scalability, usage of the old Java synchronized collections was replaced with the newer unsynchronized collections where appropriate:

- `java.util.Vector` objects were replaced with `java.util.ArrayList` objects
- `java.util.Hashtable` objects were replaced with `java.util.HashMap` objects

These internal changes should have no functionality impact.

A warning message is logged to the server log whenever a method is invoked in runtime not from the session thread since this may result in a race condition. Make sure ACE methods are never invoked from a separate thread by a Java extension.

Backward compatibility

The following XHTML Java extension APIs, are now deprecated and should be replaced as follows:

- `OnPageSubmitContext getPostData()` is now deprecated and should be replaced by `getPostDataMap()` which returns a `java.util.Map<String, String[]>` object
- `Window getAllControls()` is now deprecated and should be replaced by `getAllControlsList()` which returns a `java.util.List<XhtmlControl>` object.
- `Window getControlsByType()` is now deprecated and should be replaced by `getControlsListByType()` which returns a `java.util.List<XhtmlControl>` object.

You may continue to use the deprecated methods at the expense of creating a new Hashtable or Vector object on each invocation.

IE10 & IE11

With the introduction of IE10 Microsoft made extensive changes to adapt its browser to the same level of standards support as Firefox, Chrome and Safari. By doing so they reduced backward compatibility with previous versions. A year later, IE11 has introduced several additional changes which made it even more standards compatible and less compatible with previous versions of IE. For desktop environments, the introduction of IE11 made IE10 obsolete. However, Windows 8 based, mobile devices are still limited to using IE10.

In JIS XHTML our approach is to render pages for IE11 using the same codebase used for Firefox, Chrome and Safari. Going forward this allows us to introduce new HTML5 features into the product without duplicating our efforts on two different code bases (IE vs. non-IE). However, this implies that pages rendered by IE11 look slightly different than pages rendered using older versions of IE. IE10 is still supported using the old IE codebase. However, when working in default mode, some table and date control functionality is not supported.

Our recommendations to customers are as follows:

XHTML client

- IE11 is fully supported for both desktop and Windows Mobile devices.
- IE10 on desktop should be used in compatibility mode.
- IE10 on Windows Mobile devices can be used as long as no tables and no date controls exist.
- IE9 and earlier – we'll support these versions as long as they are supported by Micrososft but no new features will be introduced for these versions.

Java client

- In general the version of IE has little impact on the Java Plugin running the JIS Java client.
- We will support any IE version as long as it's supported by the Java Plugin itself.

Server Remote Management

JMX based tools such as Oracle's JConsole and Visual VM can be used to monitor the Java process CPU utilization, memory allocation, garbage collection and thread usage. By default, these tools can attach to Java processes running on the local machine in order to monitor its performance. However, in order to monitor a server from a remote workstation each of the Java server processes has to listen on its own remote JMX port. Specifying the JMX port using the `JavaOptions` setting of `jacadasv.ini` does not work since multiple processes will try to listen on the same port and thus fail to start.

The solution is to define the following `jacadasv.ini` settings:

```
[VMCommandLine]
JmxRemoteMonitoringStartPort=<port number>
```

The server would then increase the port number for every spawned process to prevent more than one process from allocating the same port. For example, when setting:

```
JmxRemoteMonitoringStartPort=3333
```

In a single process server the root process allocates port 3333 for JMX communication.

In a multiple process server each process will allocate its own JMX port:

```
Process 1.0 port 3333
Process 1 port 3334
Process 1.1 ports 3335
Process 1.2 ports 3336
...
```

The JMX remote connection provided by JIS does not require authentication or encryption.

Under the hood the following Java system properties are specified in the Java command line:

```
-Dcom.sun.management.jmxremote.port=[portNum]
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
```

To connect remotely to a JIS server process using the tools provided by the Java JDK:

- Install a Java JDK on the remote workstation.

To connect using Visual VM

- Execute "jvisualvm.exe" from the JDK bin folder.
- From the "File" menu choose "Add JMX Connection" and specify the address and port of the server process then click "Ok".
- A new node representing the server process is added to the "Remote" node of applications tree.
- Double click the new node to attach to the server process.

To connect using JConsole

- Execute "jconsole.exe" from the JDK bin folder.
- Click the "Remote Process" radio button; specify the address and port of the server process then click "Connect".

Note: Attaching JMX monitoring tools to a production server is generally safe since these tools are designed to add little overhead. However, some actions performed by these monitoring tools may incur additional load on the server, therefore use these tools with caution.

Remote Debugging

Warning: do not attach a debugger to a production system as this may considerably impact performance. This feature should be used exclusively for development and QA.

In order to attach a remote debugger to a server process, a unique port has to be allocated for each server process. To configure remote debugging add the following `jacadasv.ini` setting:

```
[VMCommandLine]
RemoteDebugStartPort=<port number>
```

As explained in the server remote management section, each Java process spawned by the server allocates a different port for remote debugging where the port number is increased by one for each server process. For example, when setting:

```
RemoteDebugStartPort=5005
```

In a single process server configuration the root process allocates port 5005 for remote debugging.

In a multiple process server configuration each process allocates its own remote debugging port:

```
Process 1.0 port 5005
Process 1 port 5006
Process 1.1 ports 5007
Process 1.2 ports 5008
```

Activating remote debugging

Use your favorite Java IDE remote debugging capability and attach to the specified port.

Under the hood, the Java command line parameters used by the server are:

```
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=[portNum]
```

For more information consult the documentation for your Java JRE and Java IDE.

BIN2XML Utility

One of the main challenges of integrating the JIS development environment with source control tools such as SubVersion is the in-ability to compare binary files such as the gad, acc and mnu files for each subapplication. The `bin2xml.bat` command is a simple command line utility which streams these binary files into a textual XML format so that these files can compared using external visual diff tools. To activate this utility, run `bin2xml.bat` from the `<JISRoot>` folder. Use the `-f` flag to specify the name of the file you wish to parse into an XML document. The resulting XML file has the same name as the source file with ".xml" postfix appended. For example:

```
C:\jis\bin2xml.bat -f appls\MYLIB.lib\SUBAPP.gad
…
C:\jis\appls\MYLIB.lib\SUBAPP.gad.xml created
```

Mobile Demo Web Site

The JIS mobile demo web site https://apxdemoenv.softwareag.com/ has been upgraded to V9.2.1 and the following fixes were introduced:

- Username and Password fields on the Signon screen are now using the HTML5 `<input placeholder="text">` attribute to provide place holder text.
- A back link from the Signon screen to the welcome page has been implemented.
- The dangling button problem on the main menu has been fixed.

- The table right click menu is now placed correctly.
- The site has been re-branded according to the Software GmbH official guidelines.

Logging Improvements

`LOCALIZATION` debug filter is introduced. For the Java client it has the same functionality as the localeDebugMode Applet parameter. For XHTML it provides diagnostic information related to localization resources.

JavaScript logging is now operational and additional diagnostic information is now logged both to the browser JavaScript console (if supported and enabled) and to the server log.

Detailed Description of JIS 9.2.1 Fixes

Server

#1071275 / JIS-1469 / JIS-1484 - Invoke UserDestroyApplication on unexpected disconnect

`UserDestroyApplication` is a system trigger method invoked by the server just before closing a session. Its purpose is to allow project specific code to perform last minute cleanup before a session is closed. Typically, this involves gracefully logging off the mainframe session.

In addition, `UserDestroyApplication` is now invoked in case of abnormal session termination. For example in case a Java client user closes the browser instead of closing the session window.

#5077748 / JIS-1175 / JIS-1389 - WriteUserVariable & GetUserVariables - resulting in null

`UserDestroySubApplication` is a system trigger method invoked by the server just before destroying a subapplication window when moving to another subapplication.

`UserDestroySubApplication` now has access to the destroyed subapplication variables.

Previously, when using the system triggered method `UserDestroySubApplication` to access a specific variable using an expression line or using the `GetVarValueByName` `DoMethod` the variable value was retrieved from the current subapplication not from the destroyed subapplication.

#5094537 / JIS-1285 / JIS-1331 - Running the server on Oracle Linux

The server is now able to run on an Oracle Linux platform using the same procedures used for deploying the server to a RedHat Linux platform.

JIS-1360 - JISAdminServlet requires entering user credentials twice

This has been fixed.

#5085737 / JIS-1224 / JIS-1375 - Security mismatch for password

Clear text password is no longer logged when using the `ANALYZER` filter.

#5108408 / JIS-1393 / JIS-1397 - "Unexpected Host Screen" occurs when using "RemainInScreen: True"

Screen interpreter related problem has been fixed.

JIS-1461 - Data switched between sessions when typing to the host

A rare problem of typing the wrong information to the host screen has been fixed.

#5115354 / JIS-1457 / JIS-1471 - Can't see all node in Admin console

A problem presenting node names in the administrator when deployed to WebSphere has been fixed.

JIS-1473 - Add shutdown hook when server console is disabled

When running the server as a background process on UNIX, the server -n command line flag is specified so that the server console won't block the command window.

When using this mode a shutdown hook is now registered so that shutting down the server, for example using the `jam -x shutdown` command, would terminate all server processes.

#5129821 / JIS-1551 / JIS-1561 - Directory Traversal Security problem

The embedded Jetty web server was updated to version 8.1.14 in order to resolve a security problem.

JIS-1322 - Application Verifier fixes

The application verifier now handles gracefully the following user errors:

- Application does not exist
- Application exists but has no panel files

In addition, if the application name is specified in lower or mixed case, the name is automatically converted to upper case.

#5099911 / JIS-1334 / JIS-1337 - Message handling problem

A problem in the message handling mechanism has been fixed.

#5099911 / JIS-1334 / JIS-1338 – Toolbar subapplication

A problem related to the compilation of the toolbar subapplication GS_BAR has been fixed.

#5124921 / JIS-1526 / JIS-1538 - fix infinite loop in screen interpreter

Changes to the screen interpreter internal object pool were implemented to solve infinite loop problem.

#5103216 / JIS-1358 / JIS-1359 - Checkbox translation failed when using empty settings

A problem related to the configuration of the “Localization of Checkbox Values” feature has been fixed.

#5125057 / JIS-1528 / JIS-1530 - ArrayIndexOutOfBoundsException on sendAIDKeysAndWait

Screen interpreter fix related to accessing position outside of the screen boundaries.

JIS-1477 - Root node information is not updated in Jam

The information related to processes not handling sessions is now updated in the administration utility every 60 seconds.

JIS-1570 - passing data between server and administrator

The data sent from the server to the administrator utility is now compressed by the server.

#1080114 / JIS-1553 / JIS-1558 - production server not connectable

The creation of the server sockets used for Java client communication is no longer performed in parallel to prevent problem of two server sockets allocating the same port.

#1072189 / JIS-1480 / JIS-1488 - Problem when the first session does not complete its initialization

The first session started on a server process, performs some data structure initializations. These data structures are later used by all other sessions.

Therefore, if the first session fails to initialize these data structures, other sessions fail to start. Some specific scenarios related to this problem were fixed.

#5121179 / JIS-1504 / JIS-1598 - Textbox max length set when JIS DoMethod SetVarValueByName used

Using `DoMethod SetVarValueByName` to change the value of an input field which exists only in design view also changed the field length according to the length of the value.

Java Client

#5095377 / JIS-1378 / JIS-1380 - printed host screen sometimes lose data.

The following problems were fixed:

- Distorted host screen printout
- Blinking cursor stability

#5080361 / JIS-1193 / JIS-1398 - Closing Applet

When running the client Applet inside the browser frame, terminating the session would now display the termination message inside the browser window.

#5110432 / JIS-1416 / JIS-1437 - Rollover image does not work properly on an image button

Button rollover image support was added as part of the Look & Feel enhancements for the Java client. It presents a different image when the mouse pointer moves over an image button as common in many web applications. There is no way to define the rollover image in ACE, you have to define it using a Java extension calling the `setImages()` API. The problem is that calling `setImages()` twice (once by the auto-generated code or the server and once from the code extension) resulted in either missing or incorrect rollover image. This has been fixed.

Sample code to add rollover image to a link control:

```
public abstract class ApplSubApplWindow extends
appls.JIS_1416.original.ApplSubApplWindow {

   ArrayList<GUILink> linkComps = new ArrayList<GUILink>();

   @Override
   public void setControl(Component comp, int tabIndex) {
```

```
      super.setControl(comp, tabIndex);
      if (comp instanceof GUILink) {
         linkComps.add((GUILink) comp);
      }
   }

   @Override
   public void serverDataReady() {
      super.serverDataReady();
      for (GUILink link : linkComps) {
         String image = "/classes/appls/JIS_1416/images/normal.jpg";
         String imageRollover = "/classes/appls/JIS_1416/images/
rollover.jpg";
         link.setImages(image, image, image, image, imageRollover);
      }
   }
}
```

#5115348 / JIS-1455 / JIS-1470 – Screen hanging

A deadlock related to focus management of checkbox component has been fixed.

#5138380 / JIS-1595 / JIS-1599 – Infinite refresh loop

When launching multiple sessions by auto generating the Applet tag using JavaScript, the resulting sessions may enter an infinite loop where focus is switched between sessions. This has been fixed.

JIS-1614 – Simplistic fix for a permission problem introduced by Java 8

A security exception which prevented the client from establishing socket connection to the server when using Java 1.7.0_55 and 1.8.0_05 has been fixed.

JIS-1600 – Java 1.7.0_55/1.8.0_05 fix for socket communication

A permission problem caused by setting the frame icon when using Java 1.8.0 has been fixed by disabling this functionality for unsigned code.

XHTML

Native calendar widget is now supported on Android mobile devices.

#5118397 / JIS-1486 / JIS-1487 JavaScript permission problem

Launching the JIS session from a different address than the one used for loading `<AppName>-xhtml.html` is now supported.

#5122932 / JIS-1516 / JIS-1524 Timeout Redirect

By default, when a session is closed on the server due to idle timeout, a message box is displayed to the user and the browser remains on the current page. This was identified as a security risk since leaving the browser open on the current page may expose sensitive information. On the other hand keeping the page visible allows for legitimate users to view information already entered but not submitted to the server.

Due to this trade-off we have now introduced a new configuration setting:

```
jacadasv.ini
[Http]
KeepAliveFailureAction=redirect
```

When set, the client browser will redirect to the web server root address when a session is closed by the server.

ACE

#5101893 / JIS-1349 / JIS-1350 Java Exception in Generate Runtime

A problem related to sorting XML files during generate runtime has been fixed.

#5107008 / JIS-1383 / JIS-1385 - runtime generation difference between gui and remote

Generating the runtime using the command line interface now works correctly.

#5106317 / JIS-1377 / JIS-1414 - Brown background color in color table changed to Yellow

The color table background color definition which applies to the Yellow background color was incorrectly labelled as "Brown". This has been fixed.

#5111725 / JIS-1420 / JIS-1429 - Deprecated Method used in ACE

Deprecated methods were removed from the default applicat.ion file so that new applications will no longer be able to use these methods.

Limitations of JIS 9.2.1

JIS-1594 - XHTML - Combobox - IE11 &IE10 opened above the field

When using IE10 and higher the list of items for an XHTML combobox may open above the field depending on the position of the selected item.

JIS Interface Server 9.2 Release Notes

Installation & Upgrade Information

Supported Platforms

JIS Interface Server 9.2 is supported on the following platforms:

- Windows Server 2003 Standard and Enterprise Edition (32-bit)
- Windows Server 2008 Standard and Enterprise (32-bit)
- Windows Server 2008 Standard and Enterprise (64-bit)
- Windows XP Professional (32-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (32-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (64-bit)
- Solaris SPARC 10 (64-bit)
- AIX 6.1 Power (64-bit)
- AIX 7.1 Power (64-bit)
- Red Hat Enterprise Linux 6 for x86 (64-bit)

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for Java versions, browser versions and application server versions supported by their respective vendors. Being so, when a vendor no longer supports a Java version, browser version or application server version, Software GmbH will discontinue supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows XP Professional SP3
- Windows 7

When developing on Windows 7, in order to install a JIS runtime installation on Unix, you need to add the `%windir%\System32\ftp.exe` application to the Windows firewall allowed application list.

Clients

The Java Client has been tested on the following operating systems, browser and Java versions:

- Windows 7
- RedHat Linux 6

Table 8 - 28 : JIS 9.2 Java Client Supported Browser and Java Versions

Browser	JRE
IE 8, 9	Oracle JRE 1.7.0
Firefox	Oracle JRE 1.7.0
Chrome	Oracle JRE 1.7.0

When working with a native 64-bit Windows operating system version such as Windows 7, 64-bit, in order to run the JIS Java client, the Java runtime environment must be installed twice, Java 32-bit for the 32-bit browser version and Java 64-bit for the 64-bit browser version.

The XHTML client has been tested with the following operating system and browser versions:

Table 8 - 29 : JIS 9.2 XHTML Client Supported Operating Systems and Browsers

Operating System	Browser
Windows 7	IE 8, IE 9
Windows 7	Firefox
Windows 7	Chrome
Red Hat Linux 6	Firefox
Mac OS/X	Safari
iPad IOS 6	Safari Mobile

JIS Standalone Server

The JIS standalone server has been tested in the following environments:

Table 8 - 30 : JIS 9.2 Standalone Server Supported Operating Systems

Operating Syste	Java Version
Windows 2003	Oracle 1.7.0
Windows 2008	Oracle 1.7.0
Solaris 10	Oracle 1.7.0
AIX 6.1 & 7.1	IBM 1.7.0
Red Hat Linux AS6	Oracle 1.7.0

Note: In order to run JIS on RedHat Linux 64-bit, install the shared object `libstdc++-libc6.2-2.so.3` from within the `compat-libstdc++-296-2.96-144.el6.i686.rpm` package from the RedHat Linux installation media using the command: `rpm -i compat-libstdc++-296-2.96-144.el6.i686.rpm`

J2EE Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 8 - 31 : JIS 9.2 XHTML Client Supported Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WebSphere 8.0	IBM JRE 1.6	Windows 2003
Tomcat 8	Oracale JRE 1.7	Windows 2008

OS400 Components

The DDS compiler has been tested on the following operating systems:

- OS400 V6R1
- OS400 V7R1

Retirement of Product Components

As of this release the following product features are retired:

- Innovator
- Jacada Connects
- Deploying the standalone server to AS/400
- Application Clustering

- Jacada Common Installation for application servers. For further details see "Simplified Web Application Deployment " on page 170.
- Standalone Java Client Proxy Servlet – the proxy classes are no longer distributed as part of the product installation. The only supported configuration for Java client Http/s communication is connecting directly to the server as explained in "Running the JacadaProxyServlet as part of the JIS Server" on page 263.

Note: Extended maintenance for previous versions can be requested from Software GmbH.

New Features Included in JIS 9.2

The following new features have been added for JIS 9.2

- Running the Server Using 64-bit Java
- Application Verifier
- Simplified Web Application Deployment
- Mobile Device Support
- Simplifying the Use of the LoadTest Applet
- Support for Keyboard Buffering
- Upgrading to Jetty 8.1.5
- Localization of Checkbox Values
- Client Specific Device Name Assignment
- Emulator Trace Utilities Integrated into JIS

Running the Server using 64-bit Java

Starting from this release the JIS standalone server can run using a 64-bit Java version on all 64-bit enabled operating systems and a JIS web application can be deployed to application servers running a 64-bit Java version.

As of this release, we recommend running the server with 64-bit Java.

We recommend this as 32-bit Java is limited in the amount of memory it can allocate (less than 2GB) and the number of threads it can run concurrently. These limitations effectively limit a 32-bit Java process running on a 64-bit operating system to running no more than 200 concurrent JIS sessions. This implies that your server has to sometimes spawn dozens of server processes in order to serve the desired number of concurrent sessions. In addition, JIS components which can only run in a single process such as the XHTML `RedirectionProxy` and the Java client `JISProxyServlet` are limited in the number of connections they can serve.

When using 64-bit Java the limitations on the amount of memory and the number of threads are effectively eliminated. Therefore it is possible to run more JIS sessions per process.

During our internal loadtests we found the sweet spot of sessions per process when using 64-bit Java to be between 500 and 800 sessions per process. In addition the XHTML `RedirectionProxy` and the Java client `JISProxyServlet` become more scalable and can serve thousands of concurrent sessions per server. Refer to "JIS 9.2 Appendix: Optimizing the Server for 64-bit Java Version" on page 184 for further recommendations as to how to fine tune your server configuration.

The trade-offs of using 64-bit Java vs. 32-bit Java: With regards to single session response times you shouldn't notice much difference. 64-bit Java consumes roughly 50% more memory under the same load compared to 32-bit Java, therefore if your server machine is tight on memory consider increasing the amount of physical memory installed on the server machine.

The product is still shipped with a 32-bit Java version for the following reasons: (1) some development machines still use 32-bit operating systems and therefore cannot run Java 64-bit. (2) The Application Verifier component loads 32-bit only resources in order to compare the inner workings of the new version of JIS with previous 32-bit only versions.

The ability to run the server using 64-bit required a complete rewrite in Java of the remaining server components which were written in C and were not 64-bit compatible. This internal change is one of the most significant changes made to JIS since version 9.0 has been released in 2005.

In order to run the server using a 64-bit Java version the following pre-requisites are required:

- The server machine operating system should be 64-bit enabled.
- A 64-bit enabled Java runtime environment (JRE) must be installed on the server machine.

You can download the 64-bit JRE from http://www.oracle.com/technetwork/java/javase/downloads/index.html by choosing an x64 JRE release for your server operating system.

In the `jacadasv.ini`

```
[VMCommandLine]
JavaVM=<path to the 64-bit Java executable>
```

Every application specific 3rd party component which is not part of the JIS server distribution such as JDBC drivers, external Jar files etc. must be updated to a 64-bit enabled version.

We strongly recommend running the Application Verifier and making sure it completes successfully before deploying an application to production.

Application Verifier

As a result of changes in the infrastructure, it is necessary to verify that the JIS server Screen Interpreter component produces the exact same output when given the same input in this version as in previous versions.

We recommend that you run the Verifier Utility immediately after installing the new version of JIS and report any problems back to customer support.

The Verifier works by loading screen definitions from existing application panel files (*.pnl) created during the development of the application and verifying that the C and Java code used by the previous and new version respectively both produce the exact same output. Any problems are reported in the form of an exception error report into the server log file debug_verify.log

Pre-requisites for running the Verifier:

1. Full application generate runtime must be completed successfully after installing the new version.
2. The 32-bit Java installation under `<JISRoot>\JacadaFiles\utils\jre` must be used for running the server. This is the default configuration.
3. Use the verifier in the development environment not on a runtime installation.

To activate the Verifier from the command line type the following:

```
jacadasv -v<AppName>
```

Where `<AppName>` represents the name of the application you would like to verify.

The verification process will first iterate recursively over all the application and library folders and enumerate all the panel files. It will then load the panels one by one and compare the identification results and pattern matching calculated by the new Java code to the ones calculated by the old C code. If it finds a difference it will stop running and report the problematic panel file name. More diagnostic information is printed to the `debug_verify.log` file.

The Verifier will halt when finding a problem. Report this problem to customer support and include the following information:

- debug_verify.log
- The .pnl file for which the exception was generated
- A full jpack of the application

Clear the problem condition, for example by excluding the problematic panel, and run the verification again until getting a clean run.

Additional settings defined in the `<AppName>.ini` `[ApplicationVerification]` section:

`VerificationMode` - there are two modes of verification:

1. `SCREEN` - is focused on low level comparison of the new Java code and old C code - this is the default mode, and every error in this mode should be reported to the Customer Support.
2. `SESSION` - is a higher level form of verification. When running the utility in this mode, it will generate the window components, run all methods and server side extensions and invoke all protocol messages, as well as performing the functions of the SCREEN verification type.

Note that the `SESSION` verification mode is more susceptible to false positives related to environment problems or internal assumptions implemented in project specific code, therefore it may sometimes report problems which do not really exist in the real application or even loop endlessly. Therefore it is unnecessary to report to Customer Support errors shown when running the Session verification.

ActionId - when using "`VerificationMode=SESSION`" the ActionId setting represents the action ID of the default method to invoke on the server when moving between panels. By default, the value is "18000" which represents the default action ID of the "Enter" method in ACE.

`IncludePanelFiles` - semi colon delimited list of fully qualified panel file names which need to be verified. When this setting is specified only the panels specified by the setting are loaded by the application verifier and all other application panels are ignored.

`ExcludePanelFiles` - semi colon delimited list of fully qualified panel file names which need to be excluded from the verification. When this setting is specified all panel files are recursively loaded except files specified by this setting.

If `IncludePanelFiles` is specified, `ExcludePanelFiles` is ignored.

The panel file paths specified by these settings are absolute panel file names or folder names. If folder name is specified the setting applies to all panel files under this folder and all its sub folders, the setting is case insensitive.

Example 14. JIS 9.2 Application Verifier IncludePanelFiles Setting

```
IncludePanelFiles=c:\jis\appls\MYLIB1.lib\sdf\mypanel1.pnl;c:\jis\appls\MYL
IB2.lib
```

This setting will run the verifier by loading the panel mypanel1.pnl and all .pnl files under folder MYLIB2.lib

Simplified Web Application Deployment

In previous versions of the product, deploying a JIS web application to an application server required users to first install the "common installation" component, and then specifically configure the application server to locate it. This procedure was unnecessarily complex and error prone. In addition the product administration application had to be deployed separately.

As of this release, the only required components for application server deployment are the application ear/war file generated by the Application Server runtime installation since there is no longer a common installation and administrator application.

The administration utility is now bundled inside the application ear/war file and can be launched by adding /admin to the application URL.

A new wizard step in the create runtime installation wizard selects the directory to which the configuration files are extracted and log files are written:

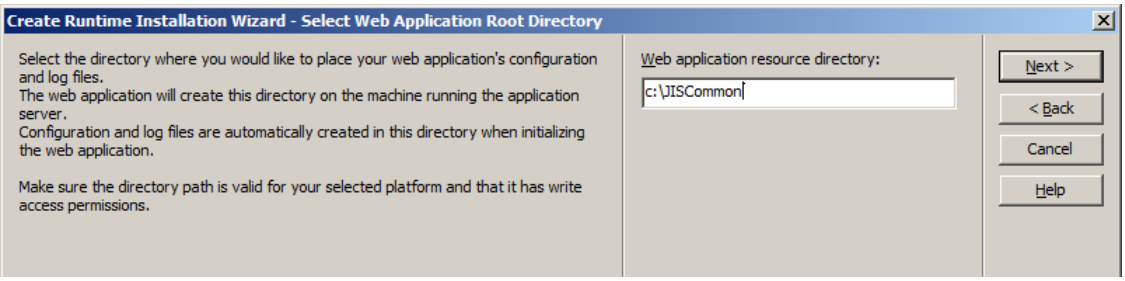

Figure 8. JIS 9.2 Create Runtime Wizard - Select Web Application Root Directory Dialog Box

In previous versions, this step was performed by the common installation. This step is optional. If the selected folder is left empty, the code will by default create the resource files and server logs in the application server's temporary folder. The Web application resource directory entered is written into the RootDir servlet parameter in the application's web.xml. During the deployment of the web application to the application server, it will create the resource folder if it does not exist and extract its configuration files from the war file to this folder.

The value of the RootDir parameter can be overridden using the Java command line property

```
-DJacadaCommonDirectory=<JIS Common directory>
```

In addition, when using ACE from the command line the following parameter was added to buildapp.xml to simulate the usage of the new wizard step:

```
TargetRootDir="<Web Application Resource Dir>"
```

Limitations

When running as a Web application, it is no longer possible to replace the runtime license from the Administrator License dialog box. Instead, replace the license file in your development environment, generate a new WAR/EAR file and deploy it to the Application Server.

Mobile Device Support

This release implements the first step towards mobile device support based on the W3C HTML5 specification. As there are still many evolving changes in this direction, we decided to start by adding support for the Apple Safari Mobile Browser and specifically target large screen iOS devices, namely the Apple iPad.

A demo application demonstrating the supported mobile features can be found in: https://apxdemoenv.softwareag.com/.

Emphasis was placed on the following:

- Safari Mobile running on iPad.
- Enabling touch capabilities to scroll within tables.
- Simulate the window and table "right click" menus using touch capabilities.
- Implement table double click using double tap functionality.
- Replace the product calendar component with the Native iPad calendar widget.

Note the following are recommended best practices when running the XHTML client using Safari Mobile on an Apple iPad:

- Using the iPad in landscape orientation is recommended rather than in the default portrait orientation.
- The built-in auto-correction feature can change the data entered incorrectly therefore it is recommended to turn off the Auto-Correction feature (Settings -> General -> Keyboard -> Auto-Correction -> OFF).
- When typing in a field on the web page using the on-screen keyboard, the most intuitive way to commit editing changes to a field is to press the <Go> button, however this button also submits the form to the server using the first <Submit> button on the form which in some cases may even be the <Exit> button. To avoid this from happening, never click the <GO> button, instead make sure you navigate between the input fields using the <Next> button and close the keyboard using the <keyboard down arrow> key.

To make the windows and table right click menus look good on the iPad, add the following code extension to ..\src\appls\<ApplName>\xhtml\Appl.java

```
public void onPageLoad(OnPageLoadContext context) {
   context.getWindow().setRMBMenuBackgroundColor("White");
```

```
    context.getWindow().setRMBMenuForegroundColor("Black");
    context.getWindow().setRMBMenuFontName("Helvetica");
    context.getWindow().setRMBMenuFontSize(12);
    context.getWindow().setRMBMenuHighlightColor("Blue");
    context.getWindow().setRMBMenuHighlightTextColor("White");
}
```

You may further adjust these settings according to your preferred application look & feel.

Limitations

- Function keys are not supported when using a mobile device, therefore if you don't have a button or Commands menu item for a specific function key you cannot execute it.
- Due to differences in the iPad screen size and the browser size and to the fact that the iPad does not show scroll bars, some screen content is not displayed though it does exist and can be reached using a drag gesture.

Simplifying the use of the LoadTest Applet

The Loadtest Applet is a component used for load testing the JIS Java and XHTML clients. This component provides the only available mechanism for load testing Java client applications and it can also be used for load testing XHTML applications.

In previous versions of the product the loadtest Applet required a Jar file which was not included in the product installation. This is no longer the case.

For detailed instructions please contact our support team.

The following improvements were made:

1. The Loadtest Applet is now packaged into the standard client JAR files so that it is no longer necessary to use the separate loadtest.jar in order to run a loadtest, or modify the classpath in order to compile the application.
2. It is no longer necessary to add application specific code to ..\user\Applet.java and ..\user\JacadaStarter.java in order to use the Applet. Every Java client application can now become a loadtest application using only Applet parameter changes and server side navigation changes.

Backward compatibility note:

Extension code that was added to ..\src\appls\<AppName>\user\Applet.java, used to initialize the loadtest applet in previous versions, has to be removed from the code since it will no longer compile and its functionality is now part of the core product.

This includes the following code in Applet.java:

```
if (isLoadtest()) {
   LoadtestController lc = new LoadtestController(this);
   add(lc, BorderLayout.CENTER);
   lc.init();
   return;
}
```

Support for Keyboard buffering

Many host emulators enable keyboard keystrokes to be buffered, so that users can continue typing without waiting for the host screen to refresh. After the host screen is refreshed, the content of the buffer is played back as if it was typed at that moment. Prior to this release, keyboard buffering required a specific runtime license. Starting from this release, this feature is an integral part of the product and no longer requires a separate license.

To enable Keyboard Buffering for the Java client:

1. 1. Open the HTML file that launches the application.
2. 2. Make sure the `UseEventDispatchThread` parameter is not set to "false", if necessary, remove the parameter or change the parameter to:

   ```
   <PARAM name="UseEventDispatchThread" value="true">
   ```
3. 3. Make sure the `EnableKeyboardBuffering` parameter is set to true (starting from this release the default value is "false"):

   ```
   <PARAM name="EnableKeyboardBuffering" value="true">
   ```
4. 4. Verify that keyboard buffering has been enabled by checking for the following debug print in the client log: "KeyboardBufferingManager is enabled"

Applet parameters related to Keyboard Buffering:

The following parameters influence how keyboard buffering functions:

Table 8 - 32 : JIS 9.2 Java Client Keyboard Buffering Parameters

Parameters	Default Value	Description
UseEventDispatchThread	`true`	Set to "`true`" to enable keyboard buffering.

Table 8 - 32 : JIS 9.2 Java Client Keyboard Buffering Parameters

Parameters	Default Value	Description
EnableKeyboardBuffering	`false`	Set to "`true`" to enable keyboard buffering. Set to "`false`" to disable keyboard buffering.
KeyboardBufferingResetKey	`Escape`	The reset key code. Pressing the specified key resets the playback buffer. Valid values can be obtained using: `KeyEvent.getKeyText(<Virtualkey code>)`
KeyboardBufferingResetKeyModifier	`Shift`	Reset key modifier. Valid codes can be obtained using: `KeyEvent.getKeyModifiersText(<Virtual modifier>)`
HideKeyboardBufferingToolbar	`true`	Set this value to "`false`" to display a toolbar which displays the status of the keyboard buffer

The following limitations influence how the keyboard buffering functions:

- Keyboard buffering should replace existing Java client code extensions which provide keyboard buffering functionality.
- Manipulating key events or focus events using Java extensions can adversely impact keyboard buffering.
- Buffering key events starts after the display of the first window.

Backward compatibility:

Customers who used the Keyboard Buffering feature in previous versions (using a runtime license) should add the following Applet parameter to enable it in this release:

```
<PARAM name="EnableKeyboardBuffering" value="true">
```

Upgrading to Jetty 8.1.5

The internal Jetty web server has been upgraded to version 8.1.5 in order to utilize the latest security and performance enhancements.

Localization of Checkbox Values

This feature affects customers converting the application in English and localizing it to various languages using the localization resource file, specifically when representing checkbox values using the host locale. For example an English Y/N checkbox can be translated into S/N in Spanish and O/N in French on the host application.

These translations are now supported using the new ini settings in the following format:

Example 15. JIS 9.2 Localization of Checkbox Values

```
[CheckboxStateTranslation]
unchecked=
checked=y/o,Y/O
intermediate=
```

The example above assumes that the checkbox is designed to represent "checked" status as "Y" while the host application represents "checked" value as "O", therefore the "checked" value in the example above is interpreted by the code as:

If the server sends "Y" translate it to "O" when sending to the host.

If the host sends "O" translate it to "Y" when sending to the server.

A similar consideration applies to the "unchecked" setting and for the "intermediate" setting when using a 3 state checkbox.

The new settings expect their value to be formatted as a comma separated list of replacement tokens similar to the "checked" setting in the example above.

Logging and Monitoring Improvements

The following improvements have been made in the message and exception logging:

- Better logging for the internal thread pool
- Improved display of the host screen in the log

- Removal of redundant debug filters
- Improved the layout of the Debug panel in the JIS Administrator
- In the JIS Administrator runtime configuration dialog the property description at the bottom of the panel now includes the name of the section and key of the corresponding ini setting.

Optimizing the Mechanism which Searches and Identifies Popup Host Screens

A host popup is a host screen which represents a popup, and whose relative location on the host screen is not fixed. Host popups are identified by their borders, regardless of their relative location on the host screen. When a host screen image sent by the host does not match a converted host screen, the screen interpreter component tries to identify it as a host popup and only if it fails it identifies it as JITGUI. Normally, host popup screens are sent by the host so that the cursor is positioned inside the popup borders. The screen interpreter uses the cursor position as a starting position for locating the popup borders. In rare cases the host popup is sent by the host with the cursor positioned outside of the popup border. In this case JIS tries to simulate various cursor positions for locating the popup borders. The drawback of this approach is that when the screen is not a host popup, valuable time is wasted.

In order to try and optimize the way popup host screens are searched for and identified when the host cursor is not positioned inside the popup window border, a new set of parameters have been added. These parameters enable defining a more exact way to search for the host popup borders on the host screen, and also offers the option not to search for popup borders at all.

The behavior is controlled by the following new `<AppName>.ini` setting in the `[ExtendedPopupBorderSearch]` section:

Enable=0/1: when set to 0 no cursor positioning will take place, and as a result a host popup is only identifiable if the cursor is inside the popup border. Set the value to 0 if your application makes extensive use of JITGUI and if you are sure every host popup includes the cursor inside its borders. 0 will be the default for new applications, otherwise the default value is 1 to maintain backward compatibility.

`StartRow` - first cursor row position (default: 5)

`RowStep` - number of characters to advance the row with each attempt (default: 7)

`StartColumn` - first cursor column position (default: 27)

`ColumnStep` - number of characters to advance the column with each attempt (default: 27)

Client Specific Device Name Assignment

This enhancement enables generating AS/400 device names which are based, for example, on the user name or computer name of the workstation running the Java client. It therefore enables associating an AS/400 device name with the workstation from which the connection originated thus enabling better traceability for the host administrator.

JIS behavior before this enhancement:

When connecting a display session (not a printer device) to an AS/400, the display session was assigned a device named `QPADEVxxxx` (where `xxxx` is a unique value assigned by the AS/400)

When connecting a display session specifying a device name, using the `LUName` ini setting, such as "`MYDEV`", then by default if the device name is not in use, it will be assigned to the session but if the device is already in use (i.e. device name collision occurred) the session will be disconnected by the host and automatically reconnected without specifying a device name thus getting a `QPADEVxxxx` device name.

The enhancement is divided into the following tasks:

1. Passing the device name from the client to the server to override the `LUName` ini setting value.
2. Defining the device name template to allow generating meaningful but unique device names based on the workstation username or computer name.
3. Gracefully handling of device name collisions.

Explanation:

1. The client can now specify the device name template using the `<PARAM name = "DeviceNameTemplate" value = "...">` Applet parameter which will override the value of the `LUName` ini setting. This allows for specific clients to specify specific device name templates based on workstation specific information.
2. In addition, when using the 5250 protocol, the user can now specify a device name template which includes wildcards (*) characters. The wildcard characters are automatically replaced by random letters or digits by the server before attempting to connect to the host thus significantly reducing the chance for device name collision.

   For example, if the client specified a device name template "`MYDEV***`" in runtime JIS will connect to the host using a device name such as `MYDEVX5D` or

MYDEVAG8 (i.e. the last 3 wildcard characters were replaced by random letters).

Furthermore if the device name template is specified by the Java client DeviceNameTemplate parameter, see 1 above. The following substitution parameters can be specified: %C+ , %C- , %U+ , %U-

The substitution parameters are composed of 3 characters:

The first character is a percent sign % indicating a substitution parameter.

The second character is a case insensitive parameter, the possible values are 'C' or 'U' meaning:

C - Computer name as specified by the COMPUTERNAME environment variable on Windows operating system.

U - User name as specified by the Java user.name system property

The third character deals with the case where the parameter value when inserted into the device name template exceeds the 10 characters device name limit:

- + : take the longest prefix of the substitute value
- - : take the longest postfix of the substitute value

Example16. JIS 9.2 DeviceNameTemplate

Assuming:

<PARAM name = "DeviceNameTemplate" value = "DEV%c+"> and computer name MCLYAF01 the device name created by the host would be DEVMCLYAF0

<PARAM name = "DeviceNameTemplate" value = "DEV%c-"> and computer name MCLYAF01 the device name created by the host would be DEVCLYAF01

3 The following 5250 enhancement http://tools.ietf.org/html/rfc4777#section-7 enables to recover from a device name collision by allowing the emulator to specify an alternate device name.

JIS now supports this emulation feature so that when a device name collision is reported by the host, JIS will modify the device name according to the device name template provided by the client or <AppName>.ini and send the modified device name to the host until finding a free device name or until the retry limit of 10 retries is exceeded. When the retry limit is exceeded, a QPADEVxxxx device name is assigned.

Limitations

When the generated device name contains characters which are not valid device name characters, the host replaces these characters with the '#' sign.

The Substitution parameters only work when using a signed version of the Java client.

Emulator Trace Utilities Integrated into JIS

The following command line utilities are now bundled as part of the JIS installation in the installation root folder:

- TracePlayer.bat (used to be Sendtrace) - plays a pre-recorded emulator trace file
- TraceProxy.bat (used to be tntrace) - records an emulator trace file for a 3rd party emulator
- TraceViewer.bat (used to be traceView) - provides visual analysis of an emulator trace file

Detailed Description of Fixes Included in JIS 9.2

Server

JIS-1263: DBCS related infrastructure changes:

1. The Java client protocol now communicates all String values using UTF8 encoding. This means that the Java version used by both client and server must support the UTF8 encoding.
2. The DBCS host screen and DBCS data is now printed correctly to the server log and can be viewed using text editors which supports UTF8 such as notepad++

When using the Java client with the Chinese language descriptor on Windows 7, you can use the following font definitions for best results:

Host screen font: `<PARAM name = "EmulatorFontName" value = "MingLiu">`

JITGUI font: `<PARAM name = "CourierFontType" value = "MingLiu">`

SI-5073798 (JIS-1156)

The default value of the following ini setting has been changed from 0 to 1 in order to provide better 5250 specification compatibility:

```
[GUISys TN5250]
SortFormatTable=1
```

SI-5076069 (JIS-1170)

Fixed disconnection which was related to the Many To One feature.

SI-5069196 (JIS-1122)

The problem that has been fixed was related to the following scenario:

1. The user worked on screen X from library A.
2. The host spontaneously sent a popup screen Y from library B so that the browser and server were then out of sync.
3. When the user submitted the form for screen X of library A, instead of getting the normal OutOfSync page, the user received an HTTP 500 response.

SI-5074062 (JIS-1155)

The server became locked when the client used IE8 or IE9 with Jetty 6.1 default configuration. This no longer occurs, as Jetty has been upgraded in this release.

JIS-1166

The `DoMethod Name` is now fully supported for all component types. It returns the name of the component as defined in ACE

JIS-1167

The `DoMethod SetModifiedFlag` on a Static receiver caused a compilation problem.

SI-1057170 (JIS-1247)

It is now possible to set the machine redirection address to be configurable. By default, when a client is redirected to a different machine or to the same machine when using the `ForceMachineRedirect=1` setting, the machine address sent to the client is the internal localhost address of the server machine.

Sometimes, such as when using network address translation, you may require that the redirect address is the external machine address and not the internal localhost address.

To specify the machine redirect address use the following `jacadasv.ini`:

```
[GeneralParameters]
MachineRedirectAddress=<machine name or ip address>
```

SI-5073514, SI-5073513 (JIS-1290)

When turning off the auto reconnect feature (`AutoReconnectToHost=0`), no message box is displayed and no session dump is created when the host closes the Telnet connection. In addition, if the same device name is reused and `RequestDefaultLUNameWhenLUNameIsRejected=0` ini setting is set, a message box is displayed but no session dump is created.

Java client

SI-5070655 (JIS-1126)

When the client communicated with the server over HTTP, data from the client was submitted more than once.

SI-5070122 (JIS-1138)

Popup windows were not printed correctly when using the print GUI feature.

SI-5076000 (JIS-1189)

When using the emulator screen, the screen sometimes was not painted correctly.

SI-5084914 (JIS-1233)

In some cases the Java internal type-ahead mechanism causes problems for the user interface. We have now introduced a setting which specifically disables this mechanism:

```
<PARAM name = "IsDisableJavaFocusManagerTypeAhead" value = "true">
```

The default value is "false" not disabling the mechanism.

Customers using the fix for ticket #5041270 should set this parameter as "true".

SI-1048440 (JIS-1237)

Creating an SSL socket from the client to the server, can be done in two ways:

1. Create a standard socket and then wrap it with an SSL socket - this is the way the product has worked so far.
2. Create an SSL socket directly - this is more mainstream approach and is introduced in this version.

A new Applet parameter enables controlling which option to use:

```
<PARAM name = "IsDirectSslSocket" value = "true">
```

The default value is "`false`", selecting the first option, maintaining backward compatibility.

The value "`true`" selects the new option.

> **Note:** SSL sockets which connect between either the proxy servlet to the server, or the server to the mainframe, are not affected by this setting.

SI-5086213 (JIS-1248)

The Properties Storage is a mechanism used for storing data locally on the client machine, such as table column arrangement. A problem with this mechanism arose when running more than one client session from the same Applet or application. Until now, each of these sessions created its own storage object on top of the singleton properties file. This way, changes made by one session were not reflected by other sessions, so that, for example, one session could overwrite changes made by other sessions. Now, the same storage instance is shared between all sessions running in the same Applet and access to the singleton storage object is synchronized. This way, a change made by one of the sessions is immediately visible to all other sessions.

SI-5071805 (JIS-1206)

The product logger no longer implements the `java.lang.Object.finalize()` method as this caused problems when running multiple sessions from the same Applet.

SI-5091782 (JIS-1286)

A problem related to printing the host screen has been fixed.

XHTML Client

SI-5085238 (JIS-1243)

When a user updated a table cell then placed the focus on the table on a protected table cell and pressed a function key, the server processed two cell changed events (1) for the changed cell (2) for the focused cell. This caused a warning about trying to update a protected cell. This scenario no longer causes a warning.

JIS-1222

When setting a value to an editable combobox table cell, the combobox value was not updated in the host.

Limitations of JIS 9.2

SI-5072258 (JIS-1135)

When the character 0x1A (Ctrl+Z) appears within the code of a BMS or MFS file, the SDF parser considers it to be an end of file character and therefore aborts the processing of the file without creating an SDF file.

Workaround: Manually delete the Ctrl+Z unprintable character from the BMS or MFS file.

SI-5077748 (JIS-1187)

Within the system triggered method `UserDestroySubApplication`, when accessing a specific variable using an expression line or using the `GetVarValueByName DoMethod`, the variable value is retrieved from the current subapplication and not from the destroyed subapplication. In order to access variables of the destroyed subapplication, users should update the subapplication variable value to a user variable while the subapplication is still alive, and then access the user variable from the `UserDestroySubApplication` system triggered method.

SI-5080361 (JIS-1197)

When using Java client with the `<PARAM name = "RunInsideBrowser" value = "true">` parameter and closing the session using the `TotalExit` method, the browser displays an empty gray border instead of displaying the actual termination message.

JIS-1260

When installing a Unix runtime using the FTP option on Windows 7, the FTP command is blocked by the Windows firewall even when FTP is enabled. You are required to either:

- Install the following Windows hot fix:
  `http://support.microsoft.com/kb/2754804`

or

- Execute the following command on the Windows workstation:
  `netsh advfirewall set global StatefulFTP disable`

JIS 9.2 Appendix: Optimizing the Server for 64-bit Java Version

Here are some best practices related to migrating the server from a 32-bit Java version to a 64-bit Java version.

> **Note:** The recommendations below are relevant only for a 64-bit Java version and not for a 32-bit Java version. If you are not sure which version of Java you are running consult the "Java Data Model" debug print in the server log.

Q: How much memory should I allocate for each server process when running using 64-bit Java?

A: In general we recommend that every JIS process will be able to allocate at least 200MB of memory. Therefore make sure to set `-mx200m` or higher for each server process including processes which are usually not serving sessions such as the root process (`debug_1.log`) and the integrator process (`debug_1.0.log`). For server processes serving sessions, the maximum amount of memory allocated to each process should be calculated as follows:

200 + 3*(number of session per process)

For example a server process running 500 sessions should be able to allocate a maximum of 200 + 3 * 500 = 1700MB of memory.

Q: What is the upper bound of memory I should allocate for server processes?

A: There are several factors to consider (1) as a rule of thumb we recommend that the total maximum memory allocated to all server processes should not exceed the total physical memory of the server machine minus 1GB. This ensures that the server never exhausts all the available machine memory which will cause swapping problems and has major effect on performance. In addition, we recommend to never allocate more than 3GB memory for a single server process since this may lead to lengthy cycles of garbage collection which may cause noticeable service interruption.

Q: How many server processes should I configure the server to spawn?

A: We recommend that the number of server processes serving sessions will be in par with the number of CPU cores used by the server machine. For example on a dual quad core server (i.e. 8 cores) we recommend setting `MaxProcess=10` and to configure the root and integrator processes not to accept sessions. This way we have 8 server processes serving sessions (1.1 - 1.8).

Q: Can you share some benchmark results?

A: On a Windows 2008 64-bit machine using dual x5670 2.93 GHZ CPU (12 cores in total) and 16GB Ram running Java 1.7.0_09 64-bit we were able to run 3,000 concurrent XHTML sessions with 35,000 screen transitions per minute. CPU utilization reached 60% and memory consumption 15GB.

The server used the following settings:

```
[GeneralParameters]
MaxProcesses=10
[Sessions]
SpareSessionsPercent=0
StartupSessionsPercent=100
MaxProcessSessions=800
MaxMachineSessions=6400
[VMCommandLine]
JavaMemory=-ms1200m -mx1200m
[M1.Level1.VMCommandLine.Server]
JavaMemory=-ms128m -mx256m
[M1.Integrator.VMCommandLine.Server]
JavaMemory=-ms128m -mx512m
[M1.Level1.Sessions]
MaxProcessSessions=0
[M1.Integrator.Sessions]
MaxProcessSessions=0
```

Note: These figures depend on many factors such as the server operating system, Java version, project specific components and mainframe response times which may be unique to your specific application, therefore we recommend load testing the server before deploying your application to production.

JIS Interface Server 9.1.2 Release Notes

Installation & Upgrade Information

Supported Platforms

JIS Interface Server 9.1.2 is supported on the following platforms:

- Windows Server 2003 Standard and Enterprise Edition (32-bit)
- Windows Server 2008 Standard and Enterprise (32-bit)
- Windows Server 2008 Standard and Enterprise (64-bit)
- Windows XP Professional (32-bit)
- Windows Vista (32-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (32-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (64-bit)
- Solaris SPARC 10 (64-bit)
- AIX 6.1 Power (64-bit)
- AIX 7.1 Power (64-bit)
- Red Hat Enterprise Linux 5 for x86 (32-bit)
- Red Hat Enterprise Linux 6 for x86 (64-bit)
- i5/OS V6R1 (OS/400)
- i5/OS V7R1 (OS/400)

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for Java versions, browser versions and application server versions supported by their respective vendors. Being so, when a vendor no longer supports a Java version, browser version or application server version, Software GmbH will discontinue supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows XP Professional SP3
- Windows 7

When developing on Windows 7, in order to install a JIS runtime installation on Unix/OS400, you need to add the `%windir%\System32\ftp.exe` application to the Windows firewall allowed application list.

Clients

The Java Client has been tested on the following operating systems, browser and Java versions:

- Windows XP Professional SP3
- Windows 7
- RedHat Linux 6

Table 9 - 33 : JIS 9.1.2 Java Client Supported Browser and Java Versions

Browser	JRE
IE 7, 8, 9	Oracle JRE 1.6.0 and 1.7.0
Firefox	Oracle JRE 1.6.0 and 1.7.0
Chrome	Oracle JRE 1.6.0 and 1.7.0

When working with a native 64-bit Windows operating system version such as Windows 7, 64-bit, in order to run the JIS Java client, the Java runtime environment must be installed twice, Java 32-bit for Internet Explorer 32-bit and Java 64-bit for Internet Explorer 64-bit.

The XHTML client has been tested with the following operating system and browser versions:

Table 9 - 34 : JIS 9.1.2 XHTML Client Supported Operating Systems and Browsers

Operating Syste	Browser
Windows 7 / XP	IE 7, IE 8, IE 9
Windows 7 / XP	Firefox
Windows 7 / XP	Safari
Windows 7 / XP	Chrome
Red Hat Linux 6	Firefox
Mac OS	Safari

JIS Standalone Server

The JIS standalone server has been tested in the following environments:

Table 9 - 35 : JIS 9.1.2 Standalone Server Supported Environments

Operating System	Java Version
Windows 2003	Oracle 1.6.0 32-bit
Windows 2008	Oracle 1.6.0 32-bit
Solaris	Oracle 1.6.0 32-bit
AIX	IBM 1.6.0 32-bit
i5/OS V6R1 and V7R11	IBM 1.6.0 32-bit
Red Hat Linux	Oracle 1.6.0 32-bit

Note: In order to run JIS on RedHat Linux 64-bit, install the shared `object libstdc++-libc6.2-2.so.3` from within the `compat-libstdc++-296-2.96-144.el6.i686.rpm` package from the RedHat Linux installation media using the command: `rpm -i compat-libstdc++-296-2.96-144.el6.i686.rpm`

J2EE Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 9 - 36 : JIS 9.1.2 XHTML Client Supported Runtime Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WebSphere 6.1.0.17	IBM JDK 1.5	Windows 2003 Enterprise Edition
WebSphere 6.1.0.17	IBM JDK 1.5	Solaris 10
WebSphere 6.1.0.17	IBM JDK 1.5	Red Hat Linux AS5
WebSphere 7.0.0.11	IBM JDK 1.6	Windows 2003 Enterprise Edition

OS400 Components

The Innovator components and the DDS compiler have been tested on the following operating systems:

- OS400 V6R1
- OS400 V7R1

Retirement of the Innovator and Studio Components

Following the announcement made in the JIS 9.1.0 Release Notes, all subsequent JIS releases and service packs will no longer provide support for the Innovator and Studio Components.

New Features Included in JIS 9.1.2

The following new features have been added for JIS 9.1.2

- Creating screen images from Natural Maps
- Simplified HTTPS/SSL Configuration
- IPv6 Support
- Specifying a Folder where the Java Client Log File will be Saved
- Logging Messages Improvements
- Proxy Servlet Improvements
- Updated JIS Perl to Version 5.12.2.0
- Session Dump Improvements
- Pattern Matching according to Character Attributes

Creating screen images from Natural Maps

One of the major strengths of JIS is its ability to create screen images directly from host screen maps. This provides many advantages over creating the screen images from screen captures. Starting from release 9.1.2, JIS now supports creating screen images directly from Software GmbH Natural map files by integrating the Software GmbH Natural parser component into the JIS codebase. Natural map files are first converted into JIS SDF standard maps and then to JIS screen images.

JIS supports creating screen images from the following Natural map formats:

- NSM format – this is the map file source itself which can be imported from the mainframe or from a NaturalOne project.
- NCD format – this map format is generated using the Natural SYSOBJH utility.

As Natural map files do not contain information required for creating function keys and popup window borders, JIS provides additional mechanisms for adding this information to the generated screen images.

Importing Natural maps

The process of creating screen images from Natural maps is very similar to the process of creating screen images from other Mainframe map formats such as BMS or MFS.

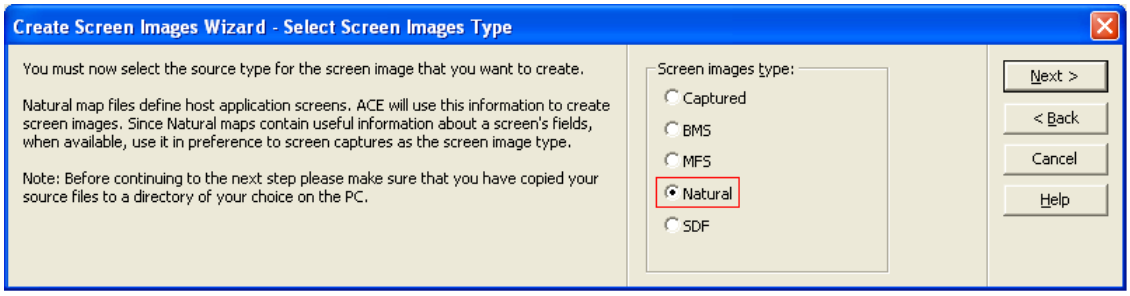

Figure 9. JIS 9.1.2 Create Runtime Images Wizard - Select Screen Images Type Dialog Box

1. In the Create Screen Images Wizard, in the Select Screen Images Type step, select Natural.
2. In the Select Source files screen, select Natural map source files to compile:
   - NCD: where each file represents one or more maps
   - NSM:
   - where each file represents a single map

Creating Popup Windows from Natural Maps

The Natural map does not contain information as to whether the map should be displayed as a popup window in runtime, and what the popup window's borders should be. Therefore, in order to support creating screen images for host popup windows the border of the window must be defined.

Popup windows in Natural are defined using the DEFINE WINDOW command: http://documentation.softwareag.com/natural/nat821mf/sm/definewi.htm. JIS uses properties similar to the ones used by the Natural DEFINE WINDOW command to display the pop-up window border in design time as close as possible as to how it would be displayed by Natural during runtime.

1. In order to achieve this, the following properties need to be specified per window map in the `natural_parser.properties` file in the `<AceRoot>` folder. Define the following properties for each window map. The existing properties file provided with the product, provides an example of the required properties:

Table 9 - 37 : JIS 9.1.2 Natural Parser Window Properties

Property Name	Description	Default Value
<Map name>.IS.WINDOW	"`TRUE`" specifies that the current map represents a window	`FALSE`
<Map name>.IS.WINDOW	This property is equivalent to the `DEFINE WINDOW` command `BASE` clause. Only the `BASE operand3/operand4` format is currently supported. The `BASE TOP/BOTTOM LEFT/RIGHT` and `BASE CURSOR` options are not supported.	1/1
<Map name>.WINDOW.SIZE	This property is equivalent to the `DEFINE WINDOW` command `SIZE` clause. The options `SIZE operand1 * operand2` and `SIZE AUTO` are supported. The `SIZE QUARTER` option is not supported.	`AUTO`
<Map name>.WINDOW.FRAME	This setting is always 3 characters long. The 1st character represents the corner character. The 2nd character represents the horizontal border and the 3rd character represents the vertical border.	Blank border
<Map name>.WINDOW.TITLE	This property is equivalent to the `DEFINE WINDOW` command `TITLE` clause.	No title

Table 9 - 37 : JIS 9.1.2 Natural Parser Window Properties

Property Name	Description	Default Value
<Map name>.WINDOW.COLOR	This property is equivalent to the `DEFINE WINDOW command FRAMED` (CD=frame-color) clause. The list of possible color values is specified in: http://documentation.softwareag.com/natural/nat821mf/parms/sp_cd.htm	The neutral color: "`NE`"

2 In the Create Screen Images Wizard, select the relevant Natural map file representing the Natural popup window content.

When creating the new subapplication using the New Subapplication wizard, the subapplication will be marked as "host popup" and the resulting host screen will include a popup border based on the window properties specified above.

Handling Function (F) Keys

By default the Natural function key lines appear in the following form on the mainframe screen:

```
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--
PF12---
      Help        Exit  Last        Flip                                Canc
```

However, many variations exist including function keys PF13 to PF24 or different function key layouts such as F3=EXIT. The various terminal commands which control the layout of the Natural function key lines are documented here:

`http://documentation.softwareag.com/natural/nat821mf/tcom/pcy.htm#PERCENT_YN`

In addition, Natural maps do not provide information about the position and layout of the Natural function key lines in runtime.

As JIS relies on the function key information in the screen image in order to identify the screen in runtime and in order to define Buttons, Menu items and Accelerator representations, it provides two different options for displaying the function key lines in the generated screen image:

1 STATIC – the default Natural function key line shown above is displayed on the screen image without the function key's description. The user needs to

capture and combine an actual host screen in order to append the function key description to the screen image. Use this mode only if your application always uses the default Natural function keys line. This mode is compatible with the screen images generated by the old mainframe based Natural parser.

2. DYNAMIC – the screen images contain prototype information and the actual function keys are created in runtime – this mode is more flexible and supports most function key layouts. Use this mode when creating a new application.

The type of function key lines displayed in the screen image is controlled by the following specific.ini setting:

```
[NaturalParser]
PFType=STATIC or PFType=DYNAMIC
```

The default value is STATIC however new applications are created with the value preset to DYNAMIC

Displaying function keys using the STATIC option

When using the STATIC option, the natural parser displays the following line exactly 2 rows from the bottom of the screen (line 22 in model 2 screens)

```
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--
PF12---
```

In runtime, JIS expects this line to appear as is, or else the screen won't be identified.

Use the Static function key layout when using Natural maps for existing projects which relied on the old mainframe based JIS Natural parser. In this scenario every map based screen image needs to be combined with a corresponding screen capture in order to overlay the function key descriptions one line from the bottom of the screen (line 23 in model 2 screens).

In order for the static keys pattern to be analyzed correctly all of the steps below must be performed:

1. Set `PFType=STATIC` as explained above before importing the Natural map.
2. Import the Natural map and compile it into a screen image using the "Create Screen Images" or "Maintain Screen Images" wizards.
3. When creating the new subapplication from the Natural screen image, in the "Select Screen Layout" step choose the "WebLookNatFKeys" screen layout or in the subapplication itself, open "Layout View" and drag the section "NatFKeys" around the function key prototypes in lines 22-23 (model 2).

```
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                                   NatFKeys
```

Figure 10. JIS 9.1.2 NatFKeys

4 Capture the screen representing the current Natural map in runtime and combine it with the subapplication created from the map in order to display the function keys together with their description.

Existing applications, using knowledge base definitions, developed in previous versions of JIS and which did not develop their own pattern definitions for matching Natural Fkeys, should append the knowledge base definitions from the NATURAL.GKB file in <JISRoot>\KB_3270 into their knowledge base list in the specific.ini (note: this is an advanced operation which requires knowledge base administration skills)

Displaying function keys using the DYNAMIC option

When using the DYNAMIC option, the Natural parser generates prototype information for the JIS Dynamic FKeys feature. During design time, JIS uses the prototypes to identify the possible FKey patterns which may appear in runtime. Therefore, there is no need to capture and combine screens.

The supported emulator commands that can be used are:

%YN, %YS, %YP – for the different layouts of the function keys line

%YA - to display both function key lines

%YF, %YL, %YX – to switch between F1-F12 and F13-F24 function key lines

The Dynamic option is the default option for new applications.

In order for the dynamic keys pattern to be analyzed correctly all of the steps below have to be performed:

1 Set `PFType=DYNAMIC` as explained above before importing the Natural map.
2 Import the Natural map and compile it into a screen image using the “Create Screen Images” or “Maintain Screen Images” wizards.
3 When creating the new subapplication from the Natural screen image, in the “Select Screen Layout” step choose the “WebLookNatDynamicFKeys” screen layout or in the subapplication itself, open “Layout View” and drag the section “NatDynamicFKeys” around the function key prototypes in lines 22-23 (model 2).

```
     PF1---            PF2---   PF3=EXIT
                                NatDynamicFKeys
     ABCDEF
```

Figure 11. JIS 9.1.2 NatDynamicFKeys

4 Existing applications, using knowledge base definitions, developed in previous versions of JIS should append the knowledge base definitions from the NATURAL.GKB file in <JISRoot>\KB_3270 into their knowledge base list

in the specific.ini (note: this is an advanced operation which requires knowledge base administration skills)

Additional Parser Configurations

Additional settings can be specified in specific.ini section [NaturalParser]

MessageLinePosition - Indicates on which line of the screen image to position the message line. The possible values are FIRST and LAST (default: LAST).

Model Type – indicates the screen model of the screen image (default: 2)

MapFileEncoding – specifies the encoding used by the Natural map source file. By default, the operating system default encoding is used.

Note: When importing maps from NaturalONE specify MapFileEncoding=UTF8

SkipWriteCommands - Select this in order not to generate maps that are called with the NATURAL WRITE command, possible values are 0 and 1 (default: 1).

Limitations

The function key lines are always displayed in their default location, two lines above the bottom of the screen (lines 22-23 with screen model 2 for example) assuming that Natural uses the terminal command %YB.

The positioning related terminal commands: %YT, %Ynn and %Y are not supported and will cause the screen not to identify in runtime.

Function key lines and message line inside a popup window, are not supported.

The message line is assumed to be in the last line of the screen (Terminal command %MB) or the first line of the screen.

Importing Natural maps from AS/400 applications is not supported.

Natural Maps and JITGUI

The default JITGUI subapplication has been enhanced to automatically recognize the Natural default function keys layouts F1-F12 and F13-F24.

Simplified HTTPS/SSL Configuration

Improving the Keystore Configuration

In order for the JIS server to use HTTPS and SSL when communicating with the client, the server has to have access to a Java keystore in which the private key and the server certificate are stored.

In previous versions the process of creating the keystore was manual using the keytool command line utility. Now, the JettyKeyStore file, the private key and a test certificate are generated automatically the first time the server is started, thus providing the ability to use HTTPS and SSL out of the box with minimal additional configuration.

Upon startup the server checks if one of the following flags is enabled in the `jacadasv.ini`:

```
[General]
JavaClientSSLEnabled=1
```

or

```
[Http]
SupportHTTPS=1
```

If so, the server looks for a file named JettyKeyStore in its classpath. If this file exists, the server loads it and uses it as its keystore. This allows users to continue to use their existing keystore or create a keystore with unique properties which cannot be created automatically.

When the file does not exist (which is always the case for a new installation), the server generates a new keystore file in the `<JISRootDir>\JacadaFiles\classes` folder with the name `JettyKeyStore` and creates an X509 server certificate based on information provided in the `jacadasv.ini [KEYSTORE]` section.

The `[KEYSTORE]` section contains a number of settings which provide the information necessary for creating an X509 certificate:

`Domain` - represents the network DNS name of the server, this should be the address provided by the client browser when connecting to the server using HTTPS or SSL. For example: `www.mydomain.com` or `sagjacada.eur.ad.sag`. Specifying this name correctly is important in order to avoid an HTTPS warning message from the client browser. By default JIS sets this value to the network name of the machine on which the server is running.

The `OrganizationalUnit`, `Organization`, `City`, `State` and `Country` settings are the X509 certificate distinguished name fields which contain free text relevant for the customer's site. The default value of each of these settings is "`Unknown`". It's important to set these settings correctly as they can be used later for generating a certificate signing request as part of the process of obtaining a valid SSL certificate instead of the auto generated test certificate.

The automatically generated `JettyKeyStore` has the following properties:

- Keystore format: `JKS`
- Keystore password: defaults to the value specified by the `jacadasv.ini` setting:

  ```
  [HTTP]
  KeystorePassword=
  ```

When the setting is not specified, the default password is "`JettyKeyStore`".

The automatically generated private key has the following properties:

- Alias: `server.key`
- Key algorithm: `RSA`
- Key password: same as the keystore password

These settings are non configurable.

The keystore is generated using the `KeyTool` utility provided by the vendor of the Java VM, currently Oracle (SUN) and IBM Java VMs are supported.

Setting up HTTPS Communication between the XHTML Client and the Server

For XHTML users, HTTPS communication is enabled once the `JettyKeyStore` has been created and the `SupportHTTPS=1` setting is defined in the `[Http]` section of `jacadasv.ini`. HTTPS communication works by default, by accessing the server on port 8443 and using the following URL:

```
https://<Server Address>:8443/<AppName>-xhtml.html
```

Note that the browser will show a browser specific warning related to the website security certificate. For example: Internet Explorer 7 will show a page titled “There is a problem with this website's security certificate.”. Ignore these warnings and continue to the web site in order to establish an HTTPS connection. For explanation of how to eliminate the warning, see “Browser Certificate Warning when Connecting to the Server” on page 200 below.

Setting up HTTPS Communication between the Java Client and the Server

Java client users, using the Applet parameter `<PARAM name = "UseHttp" value = "true">`, HTTPS is enabled once the `JettyKeyStore` has been created and the `SupportHTTPS=1` settings is defined. HTTPS communication works by default by loading the launcher HTML page from the server on port 8443 using the following URL:

```
https://<Server Address>:8443/<AppName>-signed.html
```

Setting up SSL Connection between the Java Client and the Server

Java client users, using ports communication, which is the default communication method, can now setup SSL communication without writing Java extensions.

When starting the Server, for each server process two ports will be created for SSL communication, in addition to the two existing ports used for plain text communication.

Furthermore, plain text communication can be disabled, thus forcing the user to use the SSL option.

The following `jacadasv.ini` settings control the SSL configuration:

- `JavaClientSSLEnabled` - enables the SSL communication with the Java client. Possible values: 1, 0 (default value: 1).
- `JavaClientSSLOnly` - when set, disables plain text communication, ensuring that the Java client uses SSL communication. Possible values: 1, 0 (default value: 0).
- `SSLServerPortRange` – determines the ports used for SSL communication. The default range of values that can be used for a single process configuration is 1200-1201. The port range needs to be large enough to allow each server process to allocate two SSL ports just like the allocation process for the plain text `ServerPortRange`.

By default, both SSL and plain text ports are open on the server side. To configure the Applet to use SSL communication, do one of the following:

Set the Applet parameter:

```
<PARAM name = "UseSSL" value = "true">
```

Possible values: true, false (default value: false).

Other clients not using this Applet parameter can still communicate using plain text.

Alternatively, set the `JavaClientSSLOnly=1` setting on the server side. This will force the Java client to use SSL.

Note: The following configurations will prevent the client from communicating with the server:
`JavaClientSSLEnabled=0 and <PARAM name = "UseSSL" value = "true">`
or
`JavaClientSSLOnly=1 and <PARAM name = "UseSSL" value = "false">`

Note: The communication method (Ports or HTTP/s) used by the Java client now depends only on the value of the UseHttp Applet parameter. The UsePorts Applet parameter has been deprecated. Therefore, when `<PARAM name = "UseHttp" value = "true">` is set, the client will use HTTP/S communication. Otherwise it will use the default port communication which can now be encrypted using SSL.

Note: The combination of the settings: `<PARAM name = "UseSSL" value = "true">` and `<PARAM name = "UseHttp" value = "true">` is possible but makes no sense in most configurations. It will cause the communication between the client and the proxy servlet to use HTTP or HTTPS and communication between the proxy servlet and the server to use SSL.

Browser Certificate Warning when Connecting to the Server

The `JettyKeyStore` generated automatically by the server contains an auto generated certificate which is not trusted by any official certificate authority. Therefore when connecting to the server using HTTPS, the browser will issue a warning message. There are two alternatives for eliminating the warning:

1. Manually import the server certificate into the browser. This is a browser specific procedure which tells the browser to trust the server certificate. Each browser uses its own methods for importing the certificate.
2. Generate a certificate signing request and have it signed by a certificate authority recognized by the browser and Java versions. Since JIS relies on standard Java security architecture this should be a standard process which we do not cover in this document.

SSL connection between the server and the host

It is no longer necessary to import the host certificate into the Java Keystore in order to initiate an SSL connection to a secured port defined on the host.

The following ini setting should be used in order for JIS to initiate a secure connection:

```
[GUISys TN3270] or [GUISys TN5250]
SecureHostConnection=1
```

This new setting replaces the old setting, which is currently still supported for backward compatibility.

```
SocketImplFactory=cst.server.comm.CSTSSLSocketFactory
```

IPv6 Support

JIS now supports using Internet Protocol version 6 for all runtime components including the client browser, standalone server and mainframe. All IP addresses can now be specified using the IPv6 address format.

Note: When using the XHTML RedirectionProxy and specifying server address using IPv6 address format the address must be surrounded with square brackets.

Example 17. JIS 9.1.2 RedirectionPolicy Server Address Setting

```
<Settings>
  <JacadaServerAddress>
    <IPAddress>[fe80::21c:23ff:fe31:8268]</IPAddress>
  </JacadaServerAddress>
...
<Settings>
```

Limitations

Capturing screens from ACE is not supported when the host address uses IPv6 address format.

Specifying a Folder where the Java Client Log File will be Saved

It is now possible to specify that you want to save the Client log file in a specific folder. To do this set the following Applet parameter:

```
<PARAM name ="DebugFileFolder" value="<path to a local file system folder">
```

Example 18. JIS 9.1.2 Java Client Log Setting

```
<PARAM name ="DebugFileFolder" value="c:\temp">
```

If the specified folder does not exist on the local workstation, the log file will be created in the operating system temp folder.

The Java console displays the following message indicating the location of the log file:

Client log file name is: `c:\temp\debug_1317653980569.log`

In order to use this feature you must use the signed Java client Applet.

Logging Messages Improvements

The following messages have been added to the logger:

- When receiving the version mismatch page for the Java client, the client log will now include the time stamps of the client code and server code, thus providing better understanding of the problem.
- All uncaught exceptions are now logged in the client log when using the signed Java client Applet and in the server logs.
- When running the XHTML pages inside the browser, JavaScript exceptions are now dispatched to the server and correctly logged to the server log by default.
- The server log now identifies the Linux operating system.
- The product now represents debug filters internally using a java.lang.Enum instead of the old implementation which relied on String constants.
- Therefore, when using "Method Debugging" it is required to generate the runtime again and when using code extensions which utilize debug filters, this code will need to be re-compiled. This operation is performed once, after updating the version.

Proxy Servlet Improvements

When accessing the servlet monitoring page using the URL /jisproxyservlet, the list of active connections is now printed to the server log. This can be useful in order to compare the number of open connections displayed by the proxy servlet with the number of open sessions displayed by the JIS Administrator. Open the server log and search for "List of open server connections".

Updated JIS Perl to Version 5.12.2.0

The Perl distribution used by JIS has been updated to Strawberry Perl 5.12.2.0.

Session Dump Improvements

The following improvements were made in the session dump mechanism:

1. On the Java client the dump is now printed to both the Java console and the log file.
2. On the server the dump can be turned off completely using the following `<AppName>.ini` setting:

   ```
   [SessionCoreDump]
   IsEnabled=0
   ```

3. Additional exceptions are now recorded in the dump.

Access Log

The NCSA access log contains a record of all inbound client requests that the embedded Jetty web server handles. All of the messages written to the access log are in NCSA format which is a standard format used by web servers and supported by common log analyzing tools.

The access log complements the product server log and makes it simpler to identify problems such as:

1. Response errors.
2. Slow response times.
3. Sessions jumping between servers.
4. Cookie related problems.

Example 19. JIS 9.1.2 Jetty NCSA Access Log

```
localhost 0:0:0:0:0:0:0:1 - - [13/Nov/2011:15:28:15 +0200] "GET /XHTMLV9-
xhtml.html HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/
535.2 (KHTML, like Gecko) Chrome/15.0.874.106 Safari/535.2" - 0
localhost 0:0:0:0:0:0:0:1 - - [13/Nov/2011:15:28:15 +0200] "GET /
Xhtml?JacadaApplicationName=XHTMLV9&Language=fr HTTP/1.1" 200 9142 "http://
localhost:28080/XHTMLV9-xhtml.html" "Mozilla/5.0 (Windows NT 5.1)
AppleWebKit/535.2 (KHTML, like Gecko) Chrome/15.0.874.106 Safari/535.2" -
12063
localhost 0:0:0:0:0:0:0:1 - - [13/Nov/2011:15:28:28 +0200] "GET /
XhtmlCSS?JacadaApplicationName=XHTMLV9&SessionId=1638907054&LibraryName=XHT
MLV9&SubapplName=_CSS_LOGIN&CrcCode=1288698078&JBS=8126b2c9836be51cae537e50
f369fa00299afc79777274d0 HTTP/1.1" 200 4282 "http://localhost:28080/
Xhtml?JacadaApplicationName=XHTMLV9&Language=fr" "Mozilla/5.0 (Windows NT
```

```
5.1) AppleWebKit/535.2 (KHTML, like Gecko) Chrome/15.0.874.106 Safari/535.2"
- 15
```

The lines above shows a typical sequence of requests generated when starting a new XHTML session.

The parameters logged are:

- Server name
- Client address
- Username - currently not supported
- Date and Time
- HTTP method
- URI
- Protocol version
- Response status
- Response length [bytes]
- Referrer
- User-agent
- Response time [ms]

To activate access logging, either run the server with debug level 70 or higher or add the new "ACCESS" filter to the list of debug filters.

There is one access log per server process, the log is created in the same folder as the server log. The log name is of the following format:

```
access<process_alias>_yyyy_mm_dd.log
```

For example the access log for server process 1.1 for the date November 13th, 2011 is named:

```
access_1.1_2011_11_13.log
```

The access log is rolled over every 24 hours and is kept for 14 days. There is no limit on the size of the access log. When restarting the server, the new access log is appended to the existing access log.

Pattern Matching according to Character Attributes

ACE always supported pattern matching using the following character attributes:

- Text color: foreground color of the text on the host screen.
- Background color: background color of the text on the host screen.

- Underline
- Reverse image

However, the knowledge base user interface did not support defining pattern definitions based on these attributes. Now the user interface has been added for these character attributes. The new user interface is only enabled for the “Horizontal group” and “Vertical group” pattern definition types.

A new tab was added to the "Pattern Definitions View" dialog box.

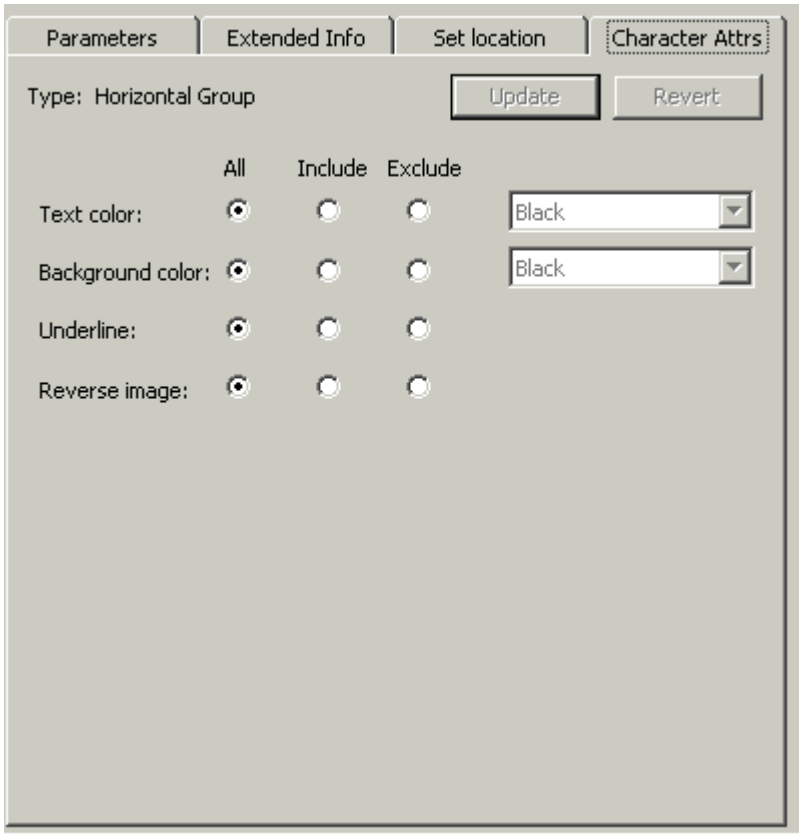

Figure 12. JIS 9.1.2 Pattern Definitions View Character Attributes Dialog Box

Each line in the dialog represents an attribute that could be set for this definition:

"All" - the definition will have no effect on pattern matching (this is the default value).

"Include" - the pattern definition will match only if the matched location on the screen includes the attribute definitions.

"Exclude" - the pattern definition will match only if the matched location on the screen does not include the attribute definitions.

The color combo boxes are disabled when the corresponding "All" radio buttons are checked.

Limitations

When matching patterns according to the "Reverse image" attribute, define "Include" "Reverse image" and "Exclude" "Underline".

Detailed Description of Fixes Included in JIS 9.1.2

> **Note:** The number at the beginning of each ticket item, represents the external support system incident or internal tracking number.

Installation

SI-1033305

When updating the operating system path, the installation now uses the time bound SendMessageTimeout Windows API instead of the SendMessage API to prevent the installation from freezing when the path cannot be updated.

Java client

JIS-647

When running the Java client as an application, the parameters equivalent to the Applet parameters in the launcher HTMLare read from the file params.txt located in the folder <JISRoot>\JacadaFiles\classes\appls\<AppName>\user. As the Applet Parameters were case insensitive and the parameters in params.txt were case sensitive, copying parameters between the two configurations led to confusing results. Now the parameters in params.txt are also case insensitive.

JIS-640

The Java client JavaDoc is now installed only when Java client is available in the CD Key

SI-5042482

The font resource loading has been fixed.

SI-5055856

Sending a client reply message starting with the value 0x0C caused the session to disconnect.

JIS-628

The default value of the UseNewHTML Applet parameter is now set to "false".

SI-5041270

The Java internal typeahead mechanism is now disabled.

SI-5059124

Since we introduced the ability to log information to the file system, by default all log messages are written to both the Java console and the client log file in the temp folder. However, some exception stack traces and the client session dump were only written to the file and not to the console. This has been fixed, and now all exceptions and the client session dump are logged both to the Java console and to the client log.

SI-5061479

When adding a custom copyright message to your application in ace.ini or ace400.ini which includes the © ASCII 0xA9 character, the © symbol was not displayed correctly in the Java client Help About dialog also causing problems localizing the copyright message.

SI-5056228

This issue is regarding the code sample in Java_Client.pdf page 270 "Methods for Controlling the Java Client Application".

Make sure that any updates to the Java client window are performed using the AWT event dispatch thread, so that the code sample on page 273 inside the run() method should look like this:

. . .

```
EventQueue.invokeAndWait(
      new Runnable() {
         public void run() {
            login.userField.setText("guest");
            login.passwordField.setText("foobar");
         }
   });
...
```

SI-5058523

All HTTP communication is now run within privileged action context.

XHTML Client

SI-5057726

In a Y/N checkbox, when the value on the host was "Y" the checkbox was still displayed as unchecked.

JIS-599

When using OptimizeStyleAttributes=1, folded tables were not displayed correctly.

JIS-1043

Line and border colors in Safari were incorrect.

JIS-1038

The page size optimization feature did not work correctly with the redirection proxy.

SI-1033559

Submitting a page while the focus was on a Combobox or on a label within a table, did not always send the correct focused control to the server.

JIS-1025

There was a problem when deploying an application to WebLogic and using page size optimization (OptimizeStyleAttributes=1).

JIS-610

In previous versions the XHTML RedirectionProxy contained Sun specific code which prevented it from using the IBM JVM. This has been fixed.

JIS-676

Deployment of J2EE application on weblogic 10.3.4 failed.

SI-501580

It is now possible to include the underscore character in the name of a table component.

SI-1044586

An exception related to using the PrintString method after calling the Close method on an external output stream has been fixed.

JIS-1102

When using the Chrome browser redundant "Keepalive" messages were sent by the browser.

SI-1034309

In a window which included a tab component, the tabbing order did not work correctly. As a result of this fix, the tab folder titles are no longer part of the tabbing order.

Server

SI-5037488

The return value of the default UserRefreshSubApplication of the NO_ATTRS screen has been changed to False, since returning True prevents the screen from refreshing, causing various problems.

SI-5036973

When using the MaintainFormatTableEntryOn5250FieldSplit ini setting, some fields were incorrectly displayed on the emulator screen.

JIS-629

In the `jacadasv.ini` file, the ProcessRespawnEnabled setting and the settings in [ProcessCheck] section are no longer supported as these caused stability and security risks.

SI-5061285

There was a memory leak related to JMX when running the server using Java 1.5. JMX is now only enabled when using Java 1.6 and above.

JIS-1075

In the JIS Administrator, the parameter transaction per minute sometimes displayed zero even though the server was actively executing transactions.

Innovator

SI-5015378

The table selection was not removed even when the table was not in focus.

Limitations of JIS 9.1.2

Limitations for JIS version 9.1.2

- When creating a screen image from any SDF, during design time, the field colors used by the color table are always considered to be Green regardless of the real field color. If you use a screen capture the color table works correctly. In runtime, the color table is correct.
- JMX is not supported when running the server using Java version 1.5.
- XHTML host printing: You need to click twice on the Connect/Disconnect button in order to connect/disconnect the printer from the host.
- When running the Java client un-signed Applet and the JISAdminServlet, an Exception related to the crossdomain.xml file is logged to the Java console.
- When running the JISAdminServlet the online help dialogs are no longer available.
- When clicking on the 'X' button to close the server console window, though the window is closed, not all server processes are terminated. We recommend that you always close the server using the QUIT command or using the JIS administrator.
- When using IE8 or higher to run a JIS XHTML application which is deployed to an application server, it is not possible to open more than one JIS session from the same browser window.
- It is not possible to run the JIS server using a 64-bit JRE. Use 32-bit JRE instead.
- The JIS common installation for J2EE deployment cannot be installed on a Windows 2008 64-bit machine.
- When running the JIS server as a Windows service, when stopping the JIS server from the administrator utility, the service is still displayed as 'started' in the Windows services panel. Stop the service from the services panel to clear out this inconsistency.

JIS Interface Server 9.1.1 Release Notes

Installation & Upgrade Information

Supported Platforms

JIS Interface Server 9.1.1 is supported on the following platforms:

- Windows Server 2003 Standard and Enterprise Edition (32-bit)
- Windows Server 2008 Standard and Enterprise (32-bit)
- Windows Server 2008 Standard and Enterprise (64-bit)
- Windows XP Professional (32-bit)
- Windows Vista (32-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (32-bit)
- Windows 7 Professional, Ultimate and Enterprise Edition (64-bit)
- Solaris SPARC 10 (64-bit)
- AIX 6.1 Power (64-bit)
- Red Hat Enterprise Linux 5 for x86 (32-bit)
- OS/400 V6R1
- OS/400 V7R1

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for Java versions, browser versions and application server versions supported by their respective vendors. Being so, when a vendor no longer supports a Java version, browser version or application server version, Software GmbH will discontinue supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows XP Professional SP3
- Windows 7 with "User Account Control" disabled

When developing on Windows 7, in order to install a JIS runtime installation on Unix/AS400, you need to add the `%windir%\System32\ftp.exe` application to the Windows firewall allowed application list.

Clients

The Java Client has been tested on the following operating systems, browser and Java versions:

- Windows XP Professional SP3
- Windows 7

Table 10 - 38 : JIS 9.1.1 Java Client Supported Browser and Java Versions

Browser	JRE
IE 7	Sun JRE 1.6.0_22
IE 8	Sun JRE 1.6.0_22
Firefox 3.5	Sun JRE 1.6.0_22

When working with a native 64-bit Windows operating system version such as Windows 7, 64-bit, in order to run the JIS Java client, the Java runtime environment must be installed twice, Java 32-bit for Internet Explorer 32-bit and Java 64-bit for Internet Explorer 64-bit.

The XHTML client has been tested with the following operating system and browser versions:

Table 10 - 39 : JIS 9.1.1 XHTML Client Supported Operating Systems and Browsers

Operating Syste	Browser
Window 7 / Windows XP	IE 7, IE 8
Window 7 / Windows XP	Firefox 3.5
Window 7 / Windows XP	Safari 5
Window 7 / Windows XP	Chrome 8
Linux x86	Firefox 3.5
Mac OS	Safari 4

JIS Standalone Server

The JIS standalone server has been tested in the following environments:

Table 10 - 40 : JIS 9.1.1 Standalone Server Supported Environments

Operating System	Java Version
Windows 2003	Sun 1.6.0_10
Windows 2008 64-bit with 32-bit JVM	Sun 1.6.0_10
Solaris 10	Sun 1.6.0_10 32-bit
AIX	IBM 1.6.0 32-bit

Table 10 - 40 : JIS 9.1.1 Standalone Server Supported Environments

Operating System	Java Version
AS/400 V6R1 and V7R1	IBM 1.6.0 32-bit
Red Hat Linux AS5	Sun 1.6.0_10 32-bit

J2EE Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 10 - 41 : JIS 9.1.1 XHTML Client Supported Runtime Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WebSphere 6.1.0.17	IBM JDK 1.5	Windows 2003 Enterprise Edition
WebSphere 6.1.0.17	IBM JDK 1.5	Solaris 10
WebSphere 6.1.0.17	IBM JDK 1.5	Red Hat Linux AS5
WebSphere 7.0.0.11	IBM JDK 1.6	Windows 2003 Enterprise Edition

OS400 Components

The DDS compiler has been tested on the following operating systems:

- OS400 V5R3
- OS400 V6R1

New Features Included in JIS 9.1.1

The following new features have been added for JIS 9.1.1

- XHTML Client Supported on Mac OS and Linux
- Server Startup Improved
- Monitoring Improvements
- Localization Improvments
- ACE
- Runtime Installation
- Java Client Improvments

XHTML

The XHTML client is now supported on Mac OS and Linux operating systems and Safari and Chrome browsers. See the "Recommended Configurations" on page 212 for specific details as to which configurations are recommended and see the "Limitations of JIS 9.1.1" on page 223 section regarding limitations when using these configurations.

> **Note:** In order to use the Mainframe function keys (`F1-F24`), when using the Safari browser on Mac OS, open the "System Preferences" dialog box, select "Keyboard" and verify that the "Use all F1,F2, etc…" checkbox is checked.

Server Changes

The Server start up, shutdown and restart times have been shortened. In order to achieve this, the following changes were made:

- The Node Registry component no longer runs as a separate Java process. Instead the Node Registry now runs within one of the other server processes. This change improves startup time and removes the need to specify the path to the `jacadasv.policy` file using the `-Djava.security.policy` flag.
- Server quit time was reduced by approximately three seconds.
- The Administrator tool now displays the Integrator (1.0) process immediately after it is fully started. Previously the Integrator process only showed up in the monitoring tool, 60 seconds after it was fully started.

Monitoring Improvements

A JIS server deployment is comprised of one or more JIS work processes. Each process represents a Java virtual machine operating system process. The JIS administrator utility is now able to provide environment and performance indicators related to the underlying Java virtual machine. Use these indicators to monitor the status of the underlying Java virtual machine.

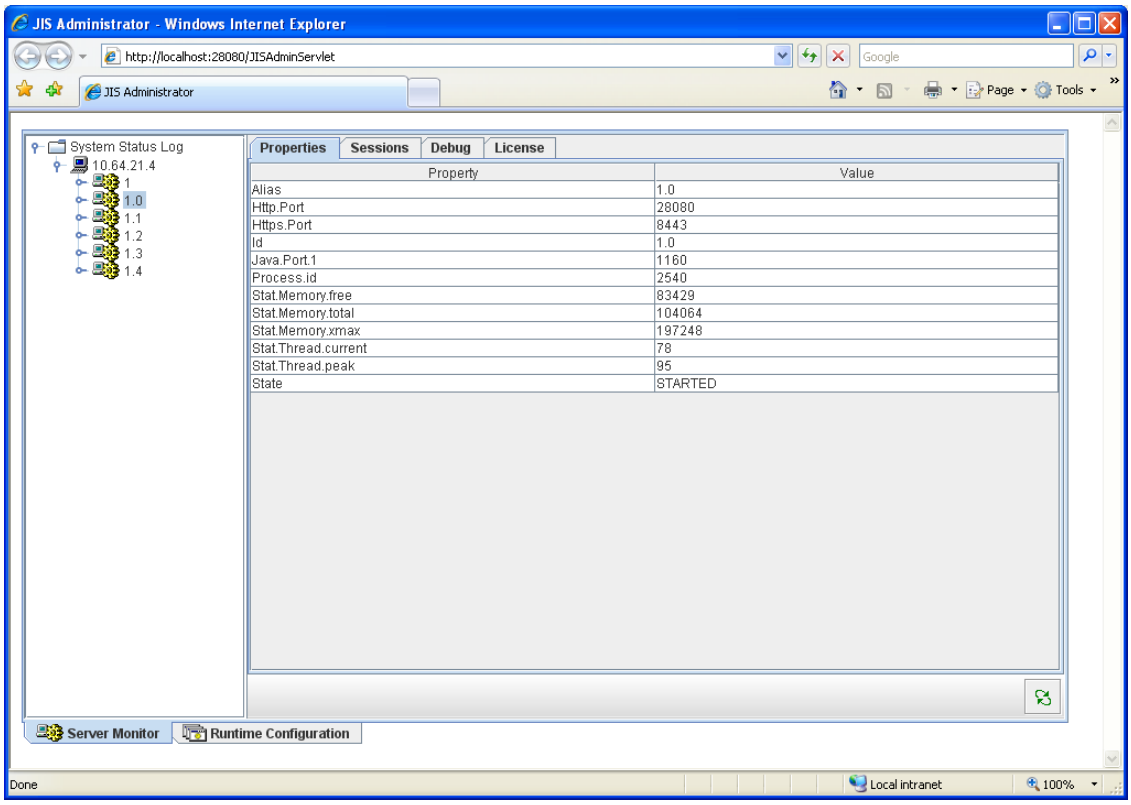

Figure 13. JIS 9.1.1 New Process Attributes

New Process Attributes

`Process.id`: Specifies the operating system process ID - useful for identifying the specific Java virtual machine process in the Windows task manager or using the Unix `ps` command.

`Stat.Memory.free` - the amount of free memory, out of the current heap memory size (specified in Kilobytes).

`Stat.Memory.total` - the current heap memory size (specified in Kilobytes).

`Stat.Memory.xmax` - the maximum allowed heap memory size, as defined by the Java -mx command line parameter (specified in Kilobytes).

`Stat.Thread.current` - the current number of operating system threads used by the Java virtual machine.

`Stat.Thread.peak` - the number of operating system threads used by the Java virtual machine at peak usage since the server started.

Best Practices

If for a given server process, `Stat.Memory.total` equals `Stat.Memory.xmax` and `Stat.Memory.free` is less than 10% of `Stat.Memory.total`, then the server process is at risk of running out of heap memory. To mitigate this risk, increase the memory heap size of the specific process using the `-mx` flag or allocate more work processes on the machine by increasing the `MaxProcesses` setting.

Before allocating more memory to a process, always make sure the server machine itself is not running out of memory.

A single Java process has limited capacity for running operating system threads, the larger the heap memory the smaller the number of threads available for the Java process. Use the following rule of thumb: if for a given server process, `Stat.Thread.peak` increases above 2000, consider allocating more Java processes on the server machine.

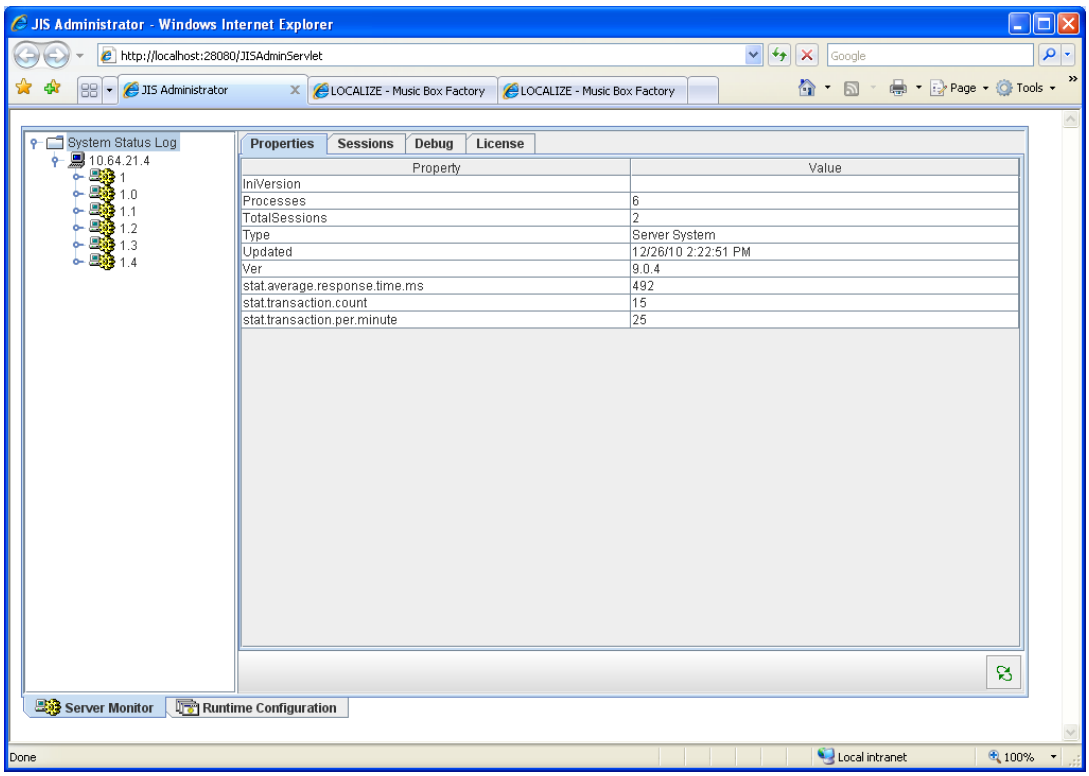

Figure 14. JIS 9.1.1 New system status attributes

New System Status Attributes

The JIS administrator now monitors the total number of transactions performed by the server at any given moment. A transaction is defined as the unit of work starting by a client action or host action and ending when the complete response is written to the client. In most cases a transaction consists of a single Mainframe screen transition.

`stat.average.response.time.ms` – the average server response time in milliseconds. The response time is measured from the time a client request was received by the server and until the response has been fully written back to the client. Therefore this value includes any think time caused by the host and the communication between the server and the host but does not include any think time caused by the client browser or communication between the client and the server. Typically a value of more than 2000 (2 seconds) indicates a performance tuning problem.

`stat.transaction.count` – the total number of transactions since the server was started. Use this parameter to evaluate the total load on the server and to make sure work is equally distributed between servers in a multi-server configuration (this parameter is not implemented when deploying the application as an .ear file)

`stat.transction.per.minute` – the current number of transactions per minute. Measuring this parameter is especially important during peak hours and during server loadtest. You can compare the value of this parameter with the Software GmbH benchmark results (this parameter is not implemented when deploying the application as an .ear file)

The new performance indicators are exposed in the following configurations:

- Standalone JAM
- JAM in J2EE
- JISAdminServlet
- JMX
- Java code acting as a JMX client

Note: A JMX enabled monitoring tool may monitor these parameters over time and allow you to chart the data and define alerts.

Localization Improvements

The following localization improvements have been made to Java Client localization support:

- A new parameter for specifying the encoding of the localization resource file has been added: "`ResourceFileEncoding`". This parameter is necessary when the encoding used when creating the resource file is different than the encoding used by the client workstation.

Example20. JIS 9.1.1 Recommended resource file encodings

To read a resource file encoded as UTF-8. This is the recommended encoding:

```
<PARAM name = "ResourceFileEncoding" value = "UTF-8">
```

To read the resource file using simplified Chinese encoding:

```
<PARAM name = "ResourceFileEncoding" value = "gbk">
```

To read the resource file using simplified Japanese encoding:

```
<PARAM name = "ResourceFileEncoding" value = "sjis">
```

- Text of dynamic menu items is now translated according to the resource file.
- Text labels in the Help-About dialog box can now be translated according to the resource file.

The following localization improvement has been made to both the Java Client and XHTML localization support:

The original string and the translated string can now include multiple appearances of the equal sign '=' and the quotes sign '"'.

ACE

Creation of the runtime installation is only possible for platforms for which runtime was generated.

Runtime Installation

It is now possible to install the JIS runtime installation on Windows to a path which includes spaces. For example: `c:\program files\<company name>\<product name>`.

This is currently not supported on Unix and AS/400.

Java Client Improvements

Mixed code warning displayed by all versions of JIS when using Java 1.6.0_19 and higher, is no longer displayed.

The `clfull-signed.jar` and `clbase-signed.jar` files are now digitally signed and time stamped; hence their signature will continue to be valid after the certificate used for signing the files has expired.

The following limitations have been removed when running the Java client as an application:

1. Link controls are now operational for activating methods (but not for opening a browser URL).
2. The `params.txt` file is no longer locked for editing while the application is running.

GUI Printing improvements

The following improvments have been made to printing from the Java Client:

- Images in popup windows are now printed correctly.
- Some deprecated APIs have been replaced and logging messages have been improved.

Detailed Description of Fixes Included in JIS 9.1.1

Server

Ticket 5024870 Generate runtime warnings are no longer displayed.

Ticket 1021672 The JBSToService utility used for creating the Windows service for JIS now prints more detailed logging information.

Ticket 5026386 Default http.xml and proxyHttp.xml files have been cleaned up.

AS/400

Ticket 5026742, 1117746 Deploying the server to AS/400 did not work properly on V5R4.

Java client

Ticket 1019009 `getLocalizedString()` API has been added.

Ticket 5028497 The `getCurrentPanel()` API may return a null value if called while a screen is being loaded. To solve this problem we introduced the `getNextPanel()` api. The `getNextPanel()` API will never return a `null` value during the session lifetime. If `genNextPanel()` is called while the screen is loading, it will wait for the screen to fully load before sending back the return value.

Ticket 5020684 When using `UseEventDispatchThread=false`, when closing popup windows, sometimes the window froze requiring restarting the browser session.

Ticket 1021759 When working with the JIS Java client proxy servlet, running inside the server and the server side closed the connection, an exception was written to the server log of the proxy servlet. This exception will no longer be written. Instead the message: "Connection closed by server" is written to the log.

XHTML Client

The default value of the `OptimizeStyleAttributes` setting has been set to 0 so that the page size optimization feature is now turned off by default.

Ticket 5020973 Cannot print spool files from the iSeries after upgrade.

Ticket 5022110 When using skin extensions, the server log displayed unimportant exceptions related to the control in focus.

ACE

Ticket 5031177 Problems relating to deleting a file during generate runtime have been fixed.

Innovator

Ticket 709592 Disconnections occurred in Dual Mode.

Ticket 708837 - Removed obsolete "`GDS C/COPY QINOSRC,INOINKLE`" statements which were added by the Translator immediately before a source line which started with a slash such as "`/eject`".

Ticket 5024661 The `RPGLE` translator will now translate source files which have a member type `SQLRPGLE`.

Limitations of JIS 9.1.1

The following are known limitations in JIS 9.1.1:

- JMX is not supported when running the server using IBM JVM version 1.5.
- Java Client: In Windows 7 and Windows 2008, menu items' keyboard shortcut text is not properly displayed.
- The JISAdminServlet and the JIS Administrator utility cannot run on Mac OS.
- The MS Sans-serif font does not look optimal on Mac OS.
- On Mac OS, Safari, the frame in Group box components, without header text, are displayed incomplete.
- When using logical colors such as "Active Border", the actual color differs between operating systems.
- XHTML host printing: You need to click twice on the Connect/Disconnect button in order to connect/disconnect the printer from the host.
- When running the Java client un-signed Applet and the JISAdminServlet, an Exception related to the crossdomain.xml file is logged to the Java console.
- When running the JISAdminServlet the online help dialogs are no longer available.
- #5008614 (#234) The runtime installation, update libraries feature does not utilize the ftp option to automatically copy files to the remote machine.
- Starting from JIS 9.1, all product components will no longer be able to run using Java 1.4 or lower versions of Java since the product code is compiled for a Java 1.5 target release. A typical error message when trying to run a product component using Java 1.4 is: Exception in thread "main" java.lang.UnsupportedClassVersionError: cst/server/module/ServerStarter (Unsupported major.minor version 49.0) (#241)
- When clicking on the 'X' button to close the server console window, though the window is closed, not all server processes are terminated. We recommend that you always close the server using the QUIT command or using the JIS administrator. (#248)
- When using IE8 to run a JIS XHTML application which is deployed to an application server, it is not possible to open more than one JIS session from the same browser window.
- It is not possible to run the JIS server using a 64-bit JRE. Use 32-bit JRE instead.
- The JIS common installation for J2EE deployment cannot be installed on a Windows 2008 64-bit machine.
- When the user updates an existing runtime installation for Unix to the same location on the Windows machine, and changes the Unix installation folder or

JVM folder using the installation wizard, the new folders are not updated into the `jacadasv.ini` file.

- When running the JIS server as a Windows service, when stopping the JIS server from the administrator utility, the service is still displayed as 'started' in the Windows services panel. Stop the service from the services panel to clear out this inconsistency.
- When running Jam on WebSphere with clustering, not all the clustered servers are displayed.

JIS Interface Server 9.1 Release Notes

Installation & Upgrade Information

JIS 9.1 does not introduce significant installation changes. If you are upgrading from JIS 9.0B or earlier versions we recommend that you read the installation instructions included in the "JIS Interface Server 9.0.3 Release Notes" on page 272.

Supported Platforms

JIS Interface Server 9.1 is supported on the following platforms:

- Windows 2003
- Windows 2008 (64-bit with 32-bit JVM)
- Solaris 10
- AIX 6.1
- AS/400 V6R1
- Red Hat Linux AS5 (32-bit)

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for operating system versions, Java versions, browser versions and application server versions supported by their respective manufacturers. Generally, when a provider stops supporting an OS version, Java version, browser version or application server version, Software GmbH will stop supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows XP Professional SP3
- Windows 7 with "User Account Control" disabled

Clients

The Java and XHTML clients have been tested on the following operating systems, browser and Java versions:

- Windows XP Professional SP3
- Windows 7

When working with a native 64-bit Windows operating system version such as Windows 7, 64-bit, in order to run the JIS Java client, the Java runtime environment must be installed twice, Java 32-bit for Internet Explorer 32-bit and Java 64-bit for Internet Explorer 64-bit.

Note: Microsoft JVM is no longer supported by JIS.

Table 11 - 42 : JIS 9.1 Java Client Supported Browser and Java Versions

Browser	JRE
IE 7	Sun JRE 1.6.0_18
IE 8	Sun JRE 1.6.0_18
Firefox 3.5	Sun JRE 1.6.0_18

The XHTML client has been tested with the following operating system and browser versions:

Table 11 - 43 : JIS 9.1 XHTML Client Supported Operating Systems and Browsers

Operating Syste	Browser
Windows XP Professions SP3	IE 7, IE 8 and Firefox 3.5
Windows 7	IE 7, IE 8 and Firefox 3.5

Standalone Server

The JIS standalone server has been tested in the following environments:

Table 11 - 44 : JIS 9.1 Standalone Server Supported Environments

Operating System	Java Version
Windows 2003	Sun 1.6.0_10
Windows 2008 64-bit with 32-bit JVM	Sun 1.6.0_10
Solaris 10	Sun 1.6.0 32-bit
AIX 6.1	IBM 1.6.0 32-bit
AS/400 V6R1	IBM 1.6.0 32-bit
Red Hat Linux **AS5 32-bit**	Sun 1.6.0 32-bit

J2EE Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 11 - 45 : JIS 9.1 XHTML Client Supported Runtime Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WebSphere 6.1.0.17	IBM JDK 1.5	Windows 2003 Enterprise Edition
WebSphere 6.1.0.17	IBM JDK 1.5	Solaris 10
WebSphere 6.1.0.17	IBM JDK 1.5	Red Hat Linux AS5
WebSphere 7.0.0.11	IBM JDK 1.6	Windows 2003 Enterprise Edition

JIS Java Proxy Servlet

The runtime of the JIS Java proxy servlet has been tested for deployment in the following environments.

Table 11 - 46 : JIS 9.1 Java Proxy Servlet Supported Environments

Application Server or Web Container	Java Runtime Environment	Operating System
JIS with embedded Jetty 6.1	Sun 1.6.0_10	Same as JIS standalone server
WebSphere 6.1.0.17	IBM JDK 1.5	Windows 2003 Enterprise Edition

OS400 Components

The Innovator components and the DDS compiler have been tested on the following operating systems:

- OS400 V5R3
- OS400 V6R1

Retirement of Product Components

As of this release, Software GmbH will be starting the retirement process of these components:

- Innovator
- Studio Components

New Features Included in JIS 9.1

The following new features have been added for JIS 9.1:

- WebSphere 7 Support
- Usability Improvements in the "Generate Runtime" and "Run Application" Wizards
- Restarting the JIS Server
- JMX Support
- XHTML Page Size Optimization Improvements
- Server Log
- Java Client Log
- XHTML JavaScript Client Log
- Localization Improvements

WebSphere 7 Support

JIS has been tested using WebSphere 7.0.0.11 on Windows 2003. Deploying a JIS application into WebSphere 7 requires additional configuration:

- Copy all the jar files from `<JISCommon>\lib` to `<WAS_HOME>\lib\ext`, this operation should be repeated every time the JIS common installation is updated.
- Add the following argument to the “Generic JVM arguments” field in the Java Virtual Machine setting panel:

`-DJacadaCommonDirectory=<Installation directory of JISCommon>`, see the attached example.

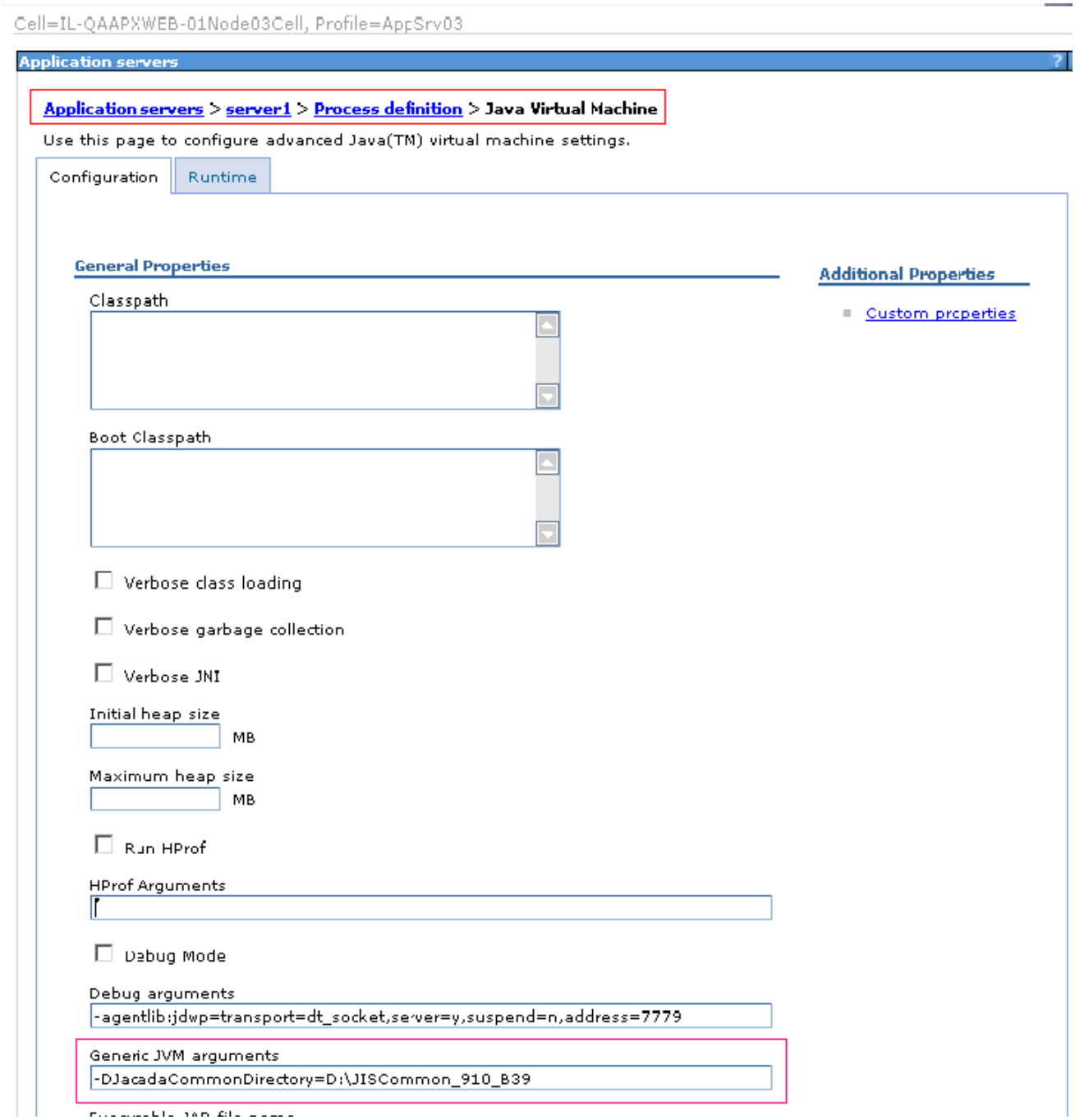

Figure 15. JIS 9.1 WebSphere 7 JVM Settings

Usability Improvements in the "Generate Runtime" and "Run Application" Wizards

The Generate Runtime functionality has been improved to allow generating the runtime while the server is running, enabling the user to continue using the existing runtime while generating a new version of the runtime.

Run Application Wizard Improvements

After clicking Finish in the last screen of the Run Application Wizard, JIS asks you whether you would like to restart the server.

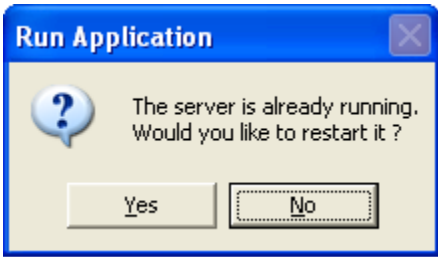

Figure 16. JIS 9.1 Application Wizard

Restarting the server enables launching the updated application in a new browser window (previously the server was not restarted and the application displayed in the browser did not reflect the changes made).

Additional improvements:

- Default compilation batch size was increased from 30 classes to 90 classes.
- The source and target release of the compiled application classes changed from 1.4 to 1.5. This allows users to write code extensions which rely on Java 5 specific syntax.
- When generating an XHTML client, the obsolete and confusing static HTML files are no longer generated in the `<JISRoot>\JacadaFiles\classes\appls\<ApplName>\xhtml\templates\original` folder. Users may delete existing old files in this folder to reduce the size of the runtime installation.

Changing Default Settings

In previous releases, after clicking Finish in the last screen of the Run Application Wizard, the default browser associated with the .html extension was opened. This approach which had several drawbacks has been abandoned. Instead, by default, the browser opened, is the browser specified in the following path "`C:\Program Files\Internet Explorer\iexplore.exe`". This path can be customized using the ini setting:

```
[RunApplicationWizard]
BrowserCommandLine=<command line for the browser application>
```

Example 21. JIS 9.1 Running The Application with Firefox

To run the application using Firefox use the following `specific.ini` setting:

```
[RunApplicationWizard]
BrowserCommandLine="C:\Program Files\Mozilla Firefox\firefox.exe" -new-
window
```

Example 22. JIS 9.1 Running the Application with Internet Explorer

To run the application using Internet Explorer 32-bit on a Windows 64-bit operating system use the following specific.ini setting:

```
BrowserCommandLine="c:\Program Files (x86)\Internet Explorer\iexplore.exe"
```

Maintaining Backward Compatibility

By default now, the server loads native resources, such as .dlr files, using a Java class loader, instead of as platform specific memory mapped files. This prevents the server from locking the resources thus allowing to generate runtime while the server is running. It is possible to change the default behavior in runtime to maintain backward compatibility. Use the following `jacadasv.ini` setting:

```
[GeneralParameters]
LoadNativeResourcesUsingJava=0
```

When using this setting you will not be able to generate runtime while the server is running.

Restarting the JIS Server

A new mechanism enables restarting the JIS server. Restarting the server is useful when updating a new version of the application or in order to reload configuration changes which require restarting the server.

This Restart command can be invoked in the following ways:

- Typing `RESTART` in the server console

```
C:\Windows\system32\cmd.exe

C:\JIS91_b29>C:\JIS91_b29\JacadaFiles\utils\jre16\bin\java -Djava.secu
y=C:\JIS91_b29\JacadaFiles\classes\jacadasv.policy -classpath C:\JIS91_
aFiles\classes;C:\JIS91_b29\JacadaFiles\classes\cst\jacadasv.jar cst.s
le.ServerStarter
Server Module
Copyright(C) 1997-2010 Software AG
Loading interface server, please wait...

Server ready: STARTED
restart
Waiting for all processes to exit
All processes exited
Started server restart
Server Module
Copyright(C) 1997-2010 Software AG
Loading interface server, please wait...

Server ready: STARTED
```

Figure 17. JIS 9.1 Restarting the JIS server via the server console

- Running the restart command via JAM's command line operations

```
C:\JIS91_b29>jam -x restart

C:\JIS91_b29>jamimpl.bat -p http://localhost:8080/port -x restart

C:\JIS91_b29>C:\JIS91_b29\JacadaFiles\utils\jre16\bin\java -Dsun.java2
-classpath C:\JIS91_b29\JacadaFiles\classes;C:\JIS91_b29\JacadaFiles\c
\jam.jar;C:\JIS91_b29\JacadaFiles\utils\javahelp\javahelp\lib\jh.jar c
AM -p http://localhost:8080/port -x restart

Administrator (JAM) Version 3.0
Copyright(C) 1997-2010 Software AG

Port file URL = http://localhost:8080/port
Restarting server. Done.

C:\JIS91_b29>
```

Figure 18. JIS 9.1 Restarting the JIS server via JAM's command line operations

- Executing the Restart command using a JMX enabled monitoring tool or from a Java class which uses JMX code

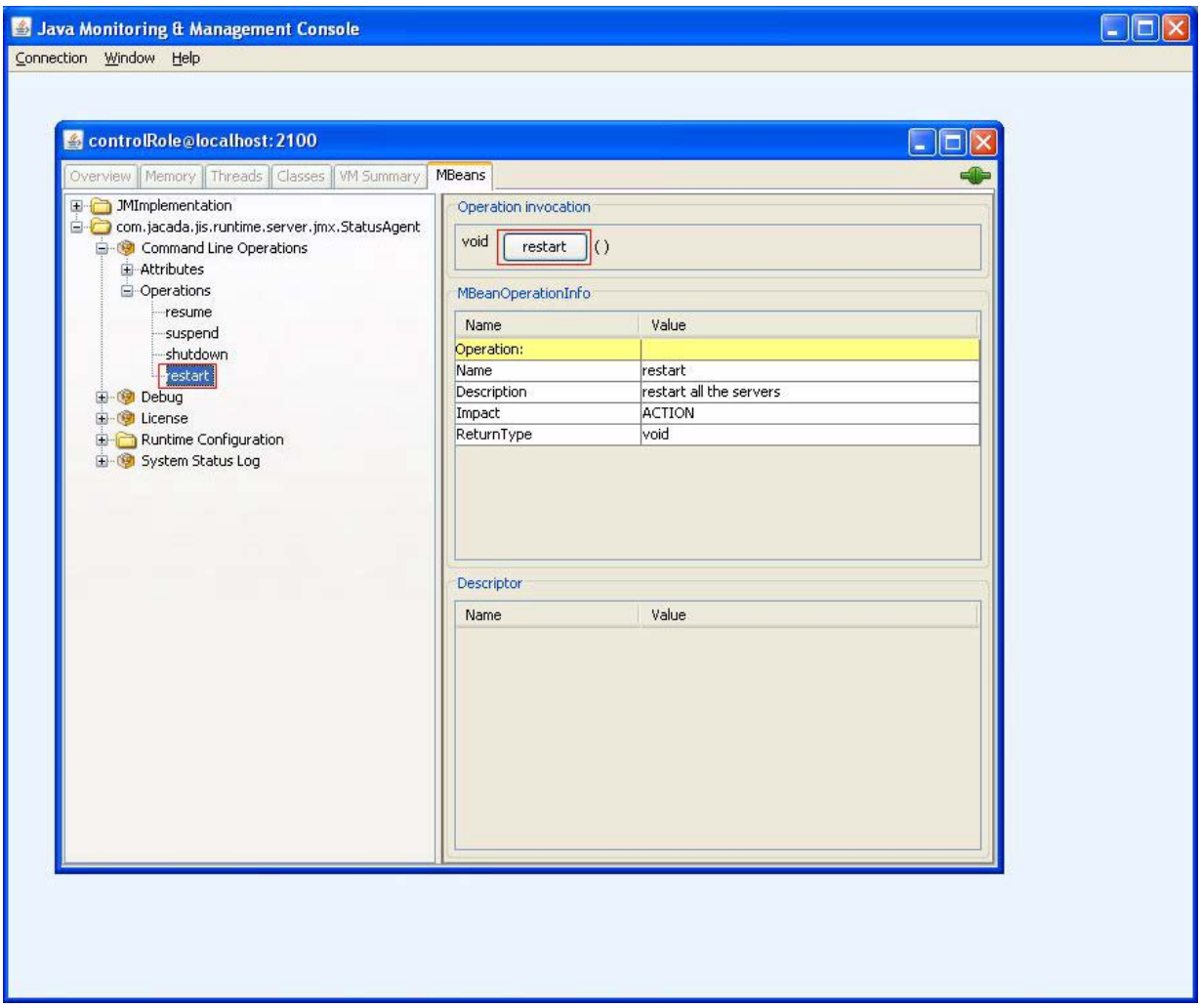

Figure 19. JIS 9.1 Restarting the JIS server via a JMX monitoring tool

JMX Support

The JMX technology provides the tools for building distributed, Web-based, modular and dynamic solutions for managing and monitoring applications. By design, this standard is suitable for adapting legacy systems, implementing new management and monitoring solutions.

JIS now enables performing administration activities including session monitoring, application configuration and server operations using JMX (previously these activities were available only via the standalone JIS Administrator tool). This allows data management (and data viewing) using monitoring tools which support JMX such as JConsole and/or by writing dedicated Java code.

In order to use JMX enable the XML server in the `jacadasv.ini` file:

```
[LogClasses]
XMLServer=

[XMLServer]
Enable=1
TimerTick=
```

> **Note:** JMX is supported only when running the standalone JIS Server, and not when using J2EE deployment.

MBeans are *managed beans*, Java objects that represent resources to be managed. Data shown and managed in the standalone JIS Administrator tool is exposed by creating matching JMX's MBeans. All the MBeans exposed by JIS are defined in the object-name root com.jacada.jis.runtime.server.log.StatusAgent and are categorized according to the data and operations they expose.

The following is a detailed list of the configuration data and administrative operations exposed using JMX:

- `System Status Log`: combines read only information for JIS servers, processes, applications and sessions. The information encompasses the same attributes shown in JIS Administrator's properties tables and is sorted in the same hierarchical tree-like topology (Root->Servers->Processes->Applications->Sessions).
- `Running Sessions`: lists information about the currently running sessions displayed as a list, and allows executing operations such as closing sessions and changing the debug level for a specific session.
- `Debug`: contains editable settings that are included in JIS Administrator's Debug panel (Debug level, Log file size, Number of log files and Log directory). The Debug MBean also exposes operations such as placing a message in the log file, clearing the log file and saving the debug settings to the ini file for future use.
- `License`: contains read-only attributes that are shown in JIS Administrator's License panel. Also allows replacing the current license file by specifying the location of a different license file.
- `Runtime configuration`: allows setting the application's ini file configuration, as done in the JIS Administrator's Runtime Configuration view. The data is sorted in a hierarchical topology - each application deployed on the server includes MBeans per each of the application's configuration sections, each of the sections includes a set of editable parameters.
- `Command line operations`: allows performing the same operations that can be invoked via JIS Administrator command line interface - Shutdown, Restart, Suspend, Resume and Status.

Connecting to the server using a JMX client application

You can connect to the JMX server using/via the client or using Java code.

Connecting via the client

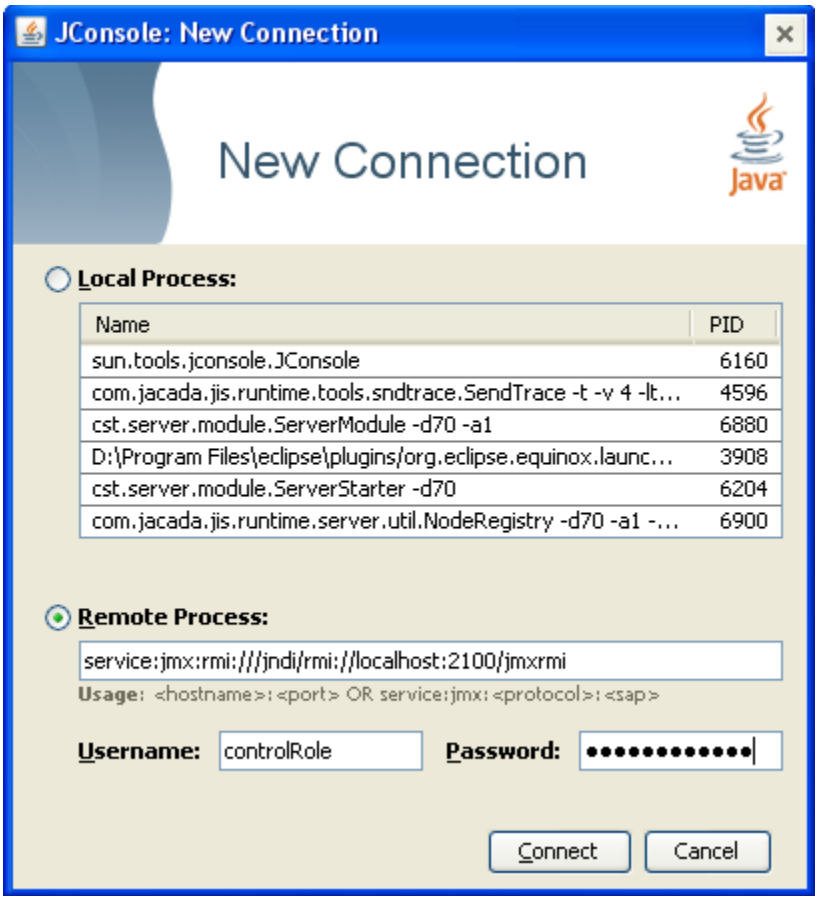

Figure 20. JIS 9.1 Connecting via Client

Log in remotely to `service:jmx:rmi:///jndi/rmi://<hostname>:<rmi port>/jmxrmi`, where `<hostname>` is the IP address or hostname of the running JIS server, and the `<rmi port>` is the port configured in the server registry node (this is the first port specified by the `jacadasv.ini [GeneralParameters] RegistryPortRange` ini setting).

Enter a username and password. Two users are defined by default: a read only user (username: `monitorRole`, password: `monitorRole`) and a user with "write" permissions (username: `controlRole`, password: `controlRole`). To change the default usernames and passwords, edit the `\classes\jmxremote.password` and `\classes\jmxremote.access` files and make the necessary changes.

Connecting using Java code:

Refer to "JIS 9.1 Appendix: JMX Code Samples" on page 248 for a number of Java code examples which demonstrate how to use JMX code to administrate the server. All examples contain pure Java code and do not rely on any product or 3rd party Jar files.

XHTML Page Size Optimization Improvements

The page size optimization feature was first introduced in JIS 9.0.4 in order to reduce the page size generated by JIS (refer to "JIS Interface Server 9.0.4 Release Notes" on page 253 for more information).

The following improvements have been made to the optimization process:

- The optimized CSS for sub-applications which contain dynamic controls, such as the JITGUI sub-application, is now generated every time the sub-application is accessed and not only the first time it is accessed.
- The optimized CSS is now generated after the server side XHTML extensions finish executing so that it reflects changes made to the page by code extensions.
- It is now possible to instruct JIS to generate a new optimized CSS for sub-applications where the page structure has been modified using a code extension. This is done by calling the `context.reOptimizeSubApplication()` api from the `onPageLoad` extension:

```
public void onPageLoad(OnPageLoadContext context) {
     ... code changes which affect the style of the specific page instance …
     context.reOptimizeSubApplication();
}
```

Server Log

The JIS server log file now uses file renaming when the current log file reaches its maximum size. Once the active log file has reached the maximum size limit, the file is renamed and the revision number is added to the file name. A new log file is created with the original name.

Example 23. JIS 9.1 Server Log Options

Start the server allowing each server process to create 6 log files of up to 100MB in size. Use the following command:

```
jacadasv -b5 -m100000000
```

The `-b5` option indicates to the server to keep 5 revisions of each process log file in addition to the current process log file.

The `-m100000000` option specifies the log file size in bytes. This defines the maximum size of a single log file to be approximately 100MB.

As a result when running the server over a period of time, the following files are created for the root process:

```
06/27/2010  05:01 PM        22,802,414 debug_1.log
06/27/2010  05:01 PM        99,999,861 debug_1.Rev1.log
06/27/2010  05:01 PM        99,999,966 debug_1.Rev2.log
06/27/2010  05:00 PM        99,999,876 debug_1.Rev3.log
06/27/2010  05:00 PM        99,999,966 debug_1.Rev4.log
06/27/2010  05:00 PM        99,999,966 debug_1.Rev5.log
```

When the size of the debug_1.log file reaches 100MB:

```
debug_1.Rev5.log is deleted
debug_1.Rev4.log is renamed to debug_1.Rev5.log
debug_1.Rev3.log is renamed to debug_1.Rev4.log
debug_1.Rev2.log is renamed to debug_1.Rev3.log
debug_1.Rev1.log is renamed to debug_1.Rev2.log
debug_1.log is renamed to debug_1.Rev1.log
```

The server continues logging into a newly created `debug_1.log` and so on.

For process 1.3, for example, the log files would be named `debug_1.3.log`, `debug_1.3.rev1.log`, ..., `debug_1.3.rev5.log`

Java Client Log

The Java Client log is now written by default to a log file named `debug_<timestamp>.log` in the `%TEMP%` folder on the local workstation and not just to the Java console. This can be controlled using the Java Applet parameter DebugFile.

The possible values are:

- `<PARAM name = "DebugFile" value = "1">` to write log messages to the Java console only, as in previous versions. The drawbacks of this setting are that the log file size is limited and there is an increase in memory consumption.
- `<PARAM name = "DebugFile" value = "2">` to write log messages only to a file in the %TEMP% folder. This approach has a drawback that only JIS log messages are written to the file and Java plugin messages are not written.

- `<PARAM name = "DebugFile" value = "3">` to write log messages to both the file and to the console (default).
- `DebugTimeStamp`: When this setting is omitted from the Applet parameters, a timestamp is added by default to the file name.
- `DebugLevel`: The existing log level 0 now provides log messages regarding errors and session dump information. A new level has been added: -1 to disable the log completely (just like debug level 0 in previous versions).

XHTML JavaScript Client Log

The XHTML client logging feature is able to log debug messages from the JavaScript used by the browser to the JIS server log.

This mechanism now has the following improvements:

- The default level is now 1 and is automatically activated (there is no longer a need to send the ClientDebugLevel URL parameter in order to activate it).
- JavaScript exceptions and their stack trace are now written to the Server Log by default.
- It is now possible to print complex messages which contain HTML text.
- It is now possible to print messages to the server log during the loading of the page.
- It is now possible to send the same message text more than once.
- The message text no longer appears in the thread name, making the text in the server log easier to read.
- Messages are written to the log in the order that they are sent from the client.

Keyboard shortcut for Java client Print GUI

The Java Client ALT+P keyboard shortcut now enables printing the active window for all windows including pop-up windows. Use the following example to customize the default keyboard shortcut. For example, the following settings will change the Print GUI shortcut key to `Ctrl+Shift+X`:

Example 24. JIS 9.1 Customizing the Default Print Keyboard Shortcut

```
<PARAM name = "PrintGuiKeyModifier" value = "Ctrl+Shift">
<PARAM name = "PrintGuiKey" value = "X">
```

Localization Improvements

Localizing Dynamic Control Strings

JIS supports localization by means of externalizing static strings defined during design time into a resource file. The process is explained in Chapter 4 of the Java client user manual. Until now the localization feature had a limitation that only static strings (i.e. strings of components which do not have data flow) in runtime were written to the resource file (StringResource.res). The current enhancement adds support for selectively writing dynamic strings of controls (i.e. strings of controls which have data flow) into the resource file. The dynamic strings that are to be written to the resource file are determined using selection rules. These rules define where to search and what to search for (using regular expressions). When the control name matches the regular expression defined in the selection rule, the control's string is written in the resource file. Note that the general localization setup and procedures were not changed by this feature.

Localization of Control Strings in Design Time¶

Dynamic control strings which match one of the selection rules are written to the StringResource.res file during the runtime generation process. The strings written are the strings which appear in design time as they appear in ACE design view.

Configuring the Selection Rules:

In the `Specific.ini [LocalizationExpressions]` section of each library, define rules to determine which control strings will be written in the resource file. Each line in this section defines a selection rule. The structure of the rule is:

```
$Key = $Value
```

Using `$Key` define the sub-application name (`$SubApplicationName`) and control type (`$ControlType`). Control types can be one of the following values: `All`, `GroupBox`, `Frame`, `TabFolder`, `DynamicGroup`, `DynamicIteration`, `PushButton`, `CheckBox`, `RadioButton`, `RadioGroup`, `CheckBox`, `OwnerDrawPushButton`, `PictureButton`, `Link`, `Static`, `Table`, `Edit`, `Window`, `Prompt`, `EditMultiline`, `Tabs`, `Menu`, `MenuItem` or `CheckboxMenuItem`.

The format of the `$Key` token can be one of the following:

- `$SubApplicationName.$ControlType=`
- `$SubApplicationName.All=`
- `$ControlType=`
- `All=`

The format of the `$Value` token is a standard Perl regular expression for matching a control's name. A comprehensive introduction to Perl regular expressions can be found here: http://perldoc.perl.org/perlre.html#regular-expressions

Order of Evaluation:

When more than one selection rule matches a control name, the order of evaluation is as follows:

- A selection rule for a specific control type in a specific sub-application takes precedence over a selection rule set for All control types in a specific sub-application.
- A selection rule for All controls in a specific sub-application takes precedence over a selection rule set for a specific control type in all sub-applications.
- The All definition (all control types in all sub-applications) is used if no other selection rule matches a control.

Note: If the regular expression defined in a selection rule didn't match the control name, the control's string will not be written to the resource file (i.e the less specific selection rule will not be evaluated for this control).

Note: All examples below assume the controls have data flow in ACE. Strings of controls without data flow are written to the resource file, no matter whether or not they match the selection rules.

Example 25. JIS 9.1 Localizing Control Strings by SubApplication and Prefix

```
[LocalizationExpressions]
LOGIN.PictureButton=^S.*
LOGIN.All=^M.*
PictureButton=^R.*
All=.*
```

In this example `PictureButton` control strings in the `LOGIN` sub-application will be written to the resource file if their name starts with the letter "`S`". All other controls in the `LOGIN` sub-application will be written to the resource file if their name starts with the letter "`M`". `PictureButton` controls in all sub-applications

(other than `LOGIN`) will be written to the resource file if their name starts with the letter "`R`". All other controls (that are not `PictureButton` type) in all other sub-applications (other than `LOGIN`) will be written to the resource file regardless of their control name.

Example 26. JIS 9.1 Localizing Control Strings by Prefix

```
[LocalizationExpressions]
All=^(?!DDS).*
```

In this example all control strings that do not start with the prefix "`DDS`" will be written to the resource file and localized in runtime according to the user's locale settings.

Localization of Table Headers

Dynamic table header strings can be written to the localization resource file without defining selection rules by using the following ini setting:

```
[JAVA]
UseStaticTableHeaders=1
```

Localization of Control Strings in Runtime

Previously, only static control strings were localized in runtime according to the user locale. Now dynamic control strings, for controls that matched one the localization expressions, are also localized.

> **Note:** The dynamic strings which are written in the resource file are only those strings that are found in the design view when generating the runtime. In runtime, these dynamic strings may change and as these values were not found previously when generating the runtime, they are not included in the resource file.

In order to add these strings to the resource file, they must be identified in runtime and recorded into the resource file manually. The identification of the strings is done by overriding the method:

```
ApplSubApplWindow.java
Public String windowMissingResource(String key) { ... }
```

The method receives the string that wasn't found in the resource file, and returns the localized version of the string. By default, the method returns the given text as is (or prefixed by "?", when working in a localization debug mode).

```
   public String windowMissingResource(String key) {
      // Write the missing string to a file or send message to the
administrator
      return key;
  }
```

Limitations

This feature is not supported in the XHTML client.

Multiplying Default Control Size by a Pre-Defined Factor

When localizing strings, the translated string is often longer than the original English string. JIS previously had limited support for changing the width of the controls by a pre-defined factor, and this support has now been expanded to also support controls without data flow.

The width factor setting affects the width of the control in the following cases:

- When creating a new control from a knowledgebase representation definition
- When creating a new control from a Floating representation definition
- When creating a new control using "Add Control" in design view
- When choosing "Adjust size by text" on an existing control in design view

This feature changes the behavior of the various control types as follows:

- Static (`Static`, `Checkbox header`) - enables sizing static controls without data flow
- Edit (`Edit`, `Prompt`, `Date`, `Combobox`, `Spin`) - no change, existing width factor already works
- Group box - no change, existing width factor already works
- Table headers - adjusts the table column width only if the table header multiplied by the width factor is wider than the table column data area. The assumption is that only table headers will be localized, while data displayed within the table data area will not be localized
- Button (`PictureButton`, `PushButton`, `Link`) - width factor is calculated based on the current button text
- Radio group - in order to determine the default component size, the width factor is multiplied by the longest Radio item
- Tab header – no change. Tab headers already have a setting similar to width factor named:

  ```
  [Converter]
  AddTabX=
  ```

The AddTabX setting can be used to increase the width of the tab header by a given number of pixels.

The width factor settings can be configured in the Window Options dialog box, in a new tab named “Control Sizing”. The “Control Sizing” tab includes previously supported control sizing parameters as well as the newly supported parameters. The new parameters include: Radio group width factor, Table header width factor and Button width factor.

The new setting "Resize components without data flow" was introduced in order to maintain backward compatibility. By default, the checkbox is unchecked so that components which have no data flow will not be resized. Select this checkbox so that the various width factors will also affect the controls which have no data flow.

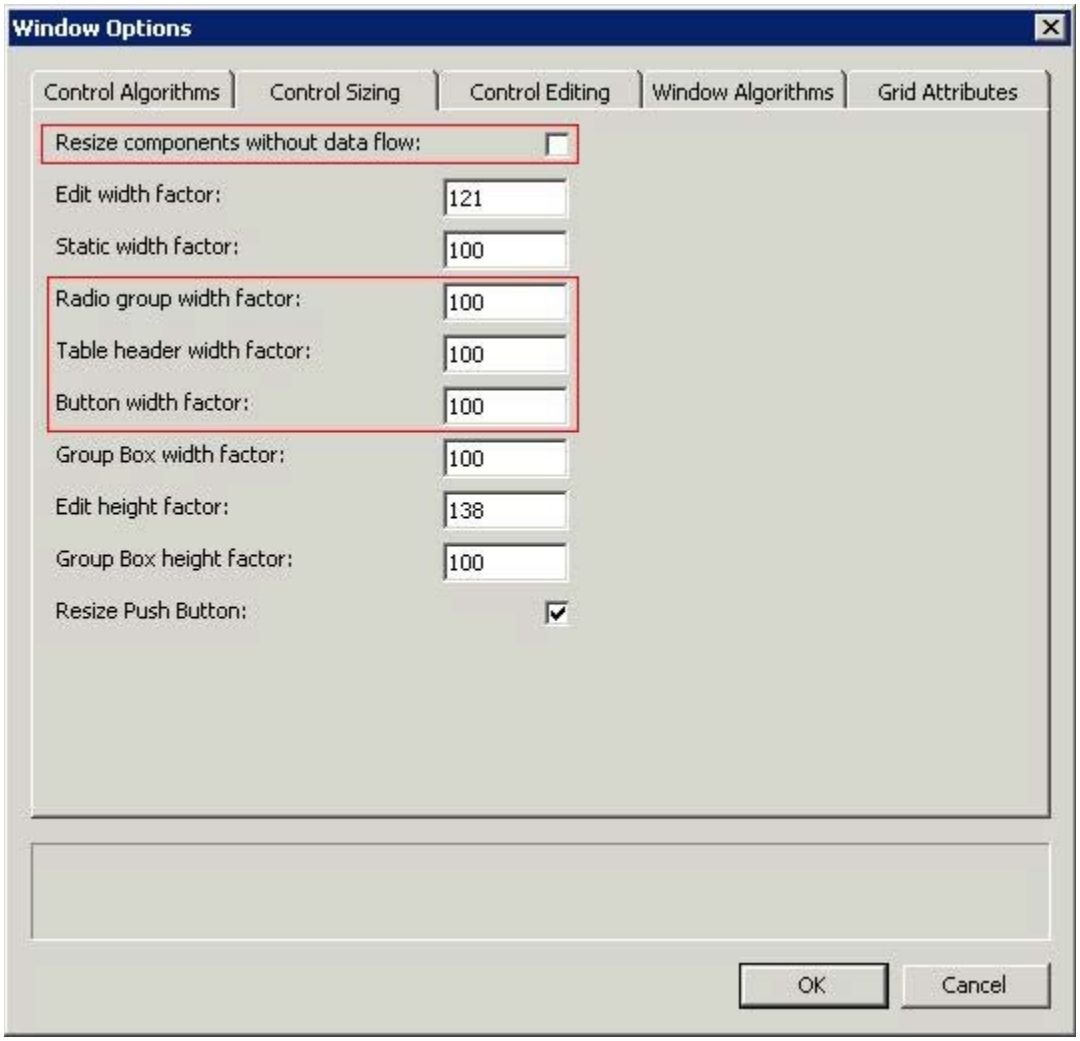

Figure 21. JIS 9.1 Window Options Control Sizing

Limitations

- The width factor settings only affects components which has no local modifications (manual or using window layout)
- The "Adjust Size By Text" function does not resize table headers
- After making changes to the "Window Options" dialog the user needs to click the <Apply> button in order for the changes to take effect

Detailed Description of Fixes Included in JIS 9.1

Installation

#5008614 The Wise script for the update libraries installation is now compiled as 32-bit executable so that it also works on 64-bit operating systems. (234)

Server

#5012282 Expanding the $CLASSPATH token did not work in Unix.

Java client

#700713 When using Keyboard Buffering, redundant key events were played back causing table cells to lose focus. (#196)

#711848 Sorting of tables did not function correctly once the table cancelSorting() API was invoked from a user extension. (#199)

#710326 The server process ID to which the client session was connected is now reported correctly in the client session dump. (#203)

#711029 When an extension code displayed a tooltip for a component located on a Swing JPanel container, an error occurred. (#222)

#1011827 Clicking on a button while the tooltip of this button was displayed caused the tooltip not to close properly. (#245)

XHTML Client

#711145 When generating runtime for sub-applications, which contain non-Latin characters, an exception is generated in runtime (Failed to transform xml to html org.xml.sax.SAXException: Fatal Error: URI=null Line=258: Character conversion error: "Malformed UTF-8 char -- is an XML encoding declaration missing?" (line number may be too low).). To prevent having this exception, change the encoding used by the generated XML files, by changing the following ini setting in `specific.ini` or in its referenced ini files:

```
[MakeExeParms]
XmlEncoding=UTF-8
```

The default value is ISO-8859-1 to maintain backward compatibility. (#197)

#711322 Due to Page Size Optimization, components were sometimes displayed incorrectly. (#216)

#711235 When merging empty combo boxes, the page was displayed incorrectly. (#217)

#5008178 Due to Page Size Optimization, cloning components from a Java extension did not work properly. (#231)

#1012135 The `<meta content="content-type" http-equiv="text/html; charset=utf-8"/>` tag was removed from the JIS page to work-around the IE8 bug described in http://blogs.msdn.com/ieinternals/archive/2009/07/27/bugs-in-the-ie8-lookahead-downloader.aspx. (#247)

#5018393 Cross Site Scripting (XSS) vulnerability related to custom error pages has been fixed. (#265)

The fallback mechanism of the localization resource files lang_state_variant used to log an exception when one of the lower levels was not found. Now if a request is issued for a resource file named EN_US and the existing resource file is just EN, no exception is logged. (#227)

#5017159 Do not write the content-length header when the response is not empty while generating the page. (258)

AS/400

#710172 The server service programs were linked to other service programs in a hardcoded library. This caused the code to fail when the hardcoded library did not exist on the customer's machine. Now, the bound service program is searched for using the library list. (#198)

Innovator

#710326 Fixed auto skip problem in Studio screens. (#204)

#690591 When using Innovator in the dual mode, the session relied on the focused window in order to determine if the current screen is innovated or green screen based. The problem was that before the main window received focus, the session was considered to be green screen even if the first window was innovated. This is fixed now. (#206)

#5011387 Clicking on the Prompt control button from a table cell focused on the incorrect row. (#254)

#5010809 The program responsible for translating messages to XML files was not converted properly to OS/400 V6R1. (#249)

Limitations of JIS 9.1

The following are known limitations of JIS 9.1:

- #5008614 (#234) The runtime installation, update libraries feature does not utilize the ftp option to automatically copy files to the remote machine.
- When generating runtime for a sub-application which contains a table component with headers which spans over more than one line, the following warning is written in the Generate Runtime console: "warning: unmappable character for encoding Cp1255". (#235)
- Starting from JIS 9.1, all product components will no longer be able to run using Java 1.4 or lower versions of Java since the product code is compiled for a Java 1.5 target release. A typical error message when trying to run a product component using Java 1.4 is: Exception in thread "main" java.lang.UnsupportedClassVersionError: cst/server/module/ServerStarter (Unsupported major.minor version 49.0) (#241)
- When clicking on the 'X' button to close the server console window, though the window is closed, not all server processes are terminated. We recommend that you always close the server using the QUIT command or using the JIS administrator. (#248)
- When upgrading a previous version of JIS, some icons are not updated to the new rebranded icons.
- When using IE8 to run a JIS XHTML application which is deployed to an application server, it is not possible to open more than one JIS session from the same browser window.
- It is not possible to run the JIS server using a 64-bit JRE. Use 32-bit JRE instead.
- The JIS common installation for J2EE deployment cannot be installed on a Windows 2008 64-bit machine.

- Though the product has been rebranded, the term "Jacada" still appears in several product entities such as directory names, urls, javadoc, method names, class names etc.
- When the user updates an existing runtime installation for Unix to the same location on the Windows machine, and changes the Unix installation folder or JVM folder using the installation wizard, the new folders are not updated into the `jacadasv.ini` file.
- When running the JIS server as a Windows service, when stopping the JIS server from the administrator utility, the service is still displayed as 'started' in the Windows services panel. Stop the service from the services panel to clear out this inconsistency.
- When running Jam on WebSphere with clustering, not all the clustered servers are displayed.

JIS 9.1 Appendix: JMX Code Samples

Following are a number of Java code examples which demonstrate how to use JMX code to administrate the server. Refer to JMX Support for further details.

All examples contain pure Java code and do not rely on any product or 3rd party Jar files.

Example 27. JIS 9.1 Changing an application's host

```
package com.jacada.jis.runtime.server.jmx.exampleScripts;

import java.io.IOException;
import java.util.HashMap;
import javax.management.*;
import javax.management.remote.*;

public class ChangeRuntimeConfigurationScript {

   public static void main(String[] args)
    throws IOException, InstanceNotFoundException,
AttributeNotFoundException,
    InvalidAttributeValueException, MBeanException, ReflectionException {

      HashMap<String, String[]> env = new HashMap<String, String[]>();
      String[] credentials = new String[] {
            "controlRole" , "controlRole"
         };
```

```
        env.put("jmx.remote.credentials", credentials);
        JMXServiceURL url = new JMXServiceURL("service:jmx:rmi:///jndi/rmi://localhost:2100/jmxrmi");
        JMXConnector jmxc = JMXConnectorFactory.connect(url, env);
        MBeanServerConnection mbsc1 = jmxc.getMBeanServerConnection();
        MBeanServerConnection mbsc = mbsc1;
        try {
            String nodeName = "Emulator Type";
            String applicationName = "NRT903";
            ObjectName objectname = new ObjectName("com.jacada.jis.runtime.server.jmx.StatusAgent:Category=Runtime Configuration, Application=" + applicationName + ",Group=" + nodeName);
            boolean isRegistered = mbsc.isRegistered(objectname);
            if (isRegistered) {
                String section = "GUISys TN5250";
                String iniSetting = "Host";
                String newValue = "ilas400";
                mbsc.setAttribute(objectname, new Attribute(section + "." + iniSetting, newValue));
                System.out.println("setting " + iniSetting + " changed to "
                                   + newValue);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            jmxc.close();
        }
    }
}
```

Example 28. JIS 9.1 Stopping the server

```
package com.jacada.jis.runtime.server.jmx.exampleScripts;

import java.io.IOException;
import java.util.HashMap;
import javax.management.*;
import javax.management.remote.*;
```

```
public class StopServerScript {

    public static void main(String[] args) {

        try {
            MBeanServerConnection mbsc = establishConnectionWithJMXServer();
            // Identifier for the command line operations
            ObjectName objectname = new
ObjectName("com.jacada.jis.runtime.server.jmx.StatusAgent:Category=Command
Line Operations");
            boolean isRegistered = mbsc.isRegistered(objectname);
            if (isRegistered) {
                // shutdown message and time in minutes
                String message = "shutting server down";
                int delay = 1;
                Object[] params = {message, delay};
                String[] signature = {"java.lang.String", "int"};
                // invoke shutdown on the server
                mbsc.invoke(objectname, "shutdown", params, signature);
              System.out.println("Shutdown invoked with message " + message
+ " will execute in " + delay + " minute/s.");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static MBeanServerConnection establishConnectionWithJMXServer()
    throws IOException {
        HashMap<String, String[]> env = new HashMap<String, String[]>();
        String[] credentials = new String[] {
            "controlRole" , "controlRole"
        };
        env.put("jmx.remote.credentials", credentials);
        JMXServiceURL url = new JMXServiceURL("service:jmx:rmi:///jndi/rmi:/
/localhost:2100/jmxrmi");
        JMXConnector jmxc = JMXConnectorFactory.connect(url, env);
        MBeanServerConnection mbsc = jmxc.getMBeanServerConnection();
        return mbsc;
    }
```

```
}
```

Example 29. JIS 9.1 Gathering data regarding running sessions

```
package com.jacada.jis.runtime.server.jmx.exampleScripts;

import java.io.IOException;
import java.util.*;
import javax.management.*;
import javax.management.remote.*;

public class ShowRunningSessionsScript {

    public static void main(String[] args) throws IOException,
    InstanceNotFoundException, AttributeNotFoundException,
    InvalidAttributeValueException, MBeanException, ReflectionException {

        StringBuffer sb = new StringBuffer();
        sb.append("Running Sessions Report\n");
        sb.append("-----------------------\n");
        HashMap<String, String[]> env = new HashMap<String, String[]>();
        String[] credentials = new String[] {
            "controlRole" , "controlRole"
        };
        env.put("jmx.remote.credentials", credentials);
        JMXServiceURL url = new JMXServiceURL("service:jmx:rmi:///jndi/rmi://
/localhost:2100/jmxrmi");
        JMXConnector jmxc1 = JMXConnectorFactory.connect(url, env);
        JMXConnector jmxc = jmxc1;
        MBeanServerConnection mbsc = jmxc.getMBeanServerConnection();
        try {
            ObjectName objectname = new
ObjectName("com.jacada.jis.runtime.server.jmx.StatusAgent:Category=Running
Sessions,Session=*");
            Set<?> mbeans = mbsc.queryMBeans(objectname, null);
            sb.append(mbeans.size() + " sessions are running.\n");
            sb.append("\n");
            Iterator<?> iter = mbeans.iterator();
            while (iter.hasNext()) {
```

```
                ObjectInstance oi = (ObjectInstance)iter.next();
                ObjectName on = oi.getObjectName();
              sb.append("Session id " + on.getKeyProperty("Session")+"\n");
                sb.append("-----------------\n");
                sb.append(mbsc.invoke(on, "ListAttributes", null,
null)+"\n");
            }
            System.out.println(sb.toString());
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            jmxc.close();
        }
    }
}
```

JIS Interface Server 9.0.4 Release Notes

Installation & Upgrade Information

JIS 9.0.4 is the next service pack of the JIS 9.0 major release. JIS 9.0.4 does not introduce significant installation changes compared to JIS 9.0.3. If however you are upgrading from JIS 9.0B or earlier versions we recommend that you read the installation instructions included in the "JIS Interface Server 9.0.3 Release Notes" on page 272.

Supported Platforms

JIS Interface Server 9.0.4 is supported on the following platforms:

- Windows 2003
- Windows 2008 (64-bit with 32-bit JVM)
- Solaris 10
- AIX 6.1
- AS/400 V6R1

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for operating system versions, Java versions, browser versions and application server versions supported by their respective manufacturers. Generally, when a provider stops supporting an OS version, Java version, browser version or application server version, Software GmbH will stop supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows XP Professional SP3
- Windows Vista with "User Account Control" disabled

Clients

The Java Client has been tested on the following operating systems, browser and Java versions:

- Windows XP Professional SP3
- Windows Vista

Table 12 - 47 : JIS 9.0.4 Java Client Supported Browser and Java Versions

Browser	JRE
IE 7	Sun JRE 1.6.0
IE 8	Sun JRE 1.6.0
Firefox 3.0	Sun JRE 1.6.0
Firefox 3.5	Sun JRE 1.6.0

The XHTML client has been tested with the following operating system and browser versions:

Table 12 - 48 : JIS 9.0.4 XHTML Client Supported Operating Systems and Browsers

Operating Syste	Browser
Windows XP Professional SP3	IE 7, IE 8, Firefox 3.0 and Firefox 3.5
Windows Vista	IE 7, IE 8, Firefox 3.0 and Firefox 3.5

JIS Standalone Server

The JIS standalone server has been tested in the following environments:

Table 12 - 49 : JIS 9.0.4 Standalone Server Supported Environments

Operating System	Java Version
Windows 2003	Sun 1.6.0_10
Windows 2008 64-bit with 32-bit JVM	Sun 1.6.0_10
Solaris 10	Sun 1.6.0 32-bit
AIX 6.1	IBM 1.6.0 32-bit
AS/400 V6R1	IBM 1.6.0 32-bit

J2EE Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 12 - 50 : JIS 9.0.4 XHTML Client Supported Runtime Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WeboLogic 8.1 SP6	JRocket 1.4.2	Windows 2003 Enterprise Edition
WebSphere 6.1.0_17	IBM JDK 1.5	Windows 2003 Enterprise Edition
WebSphere 6.1.0_17	IBM JDK 1.5	Solaris 10

Table 12 - 50 : JIS 9.0.4 XHTML Client Supported Runtime Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WebSphere 6.1.0_17	IBM JDK 1.5	Red Hat Linux AS4

JIS Java Proxy Servlet

The runtime of the JIS Java proxy servlet has been tested for deployment in the following environments.

Table 12 - 51 : JIS 9.0.4 Java Proxy Servlet Supported Environments

Application Server or Web Container	Java Runtime Environment	Operating System
JIS with embedded Jetty 6.1	Sun 1.6.0_10	Same as JIS standalone server
WebSphere 6.1.0_17	IBM JDK 1.5	Windows 2003 Enterprise Edition

AS400 Components

The Innovator components and the DDS compiler have been tested on the following operating systems:

- OS400 V5R3
- OS400 V6R1

Installing JIS 9.0.4 server on OS400 V6R1

Preparing your OS400 V6R1 machine Java environment

1. Ensure that Java 1.6 32-bit JRE is installed and set as the default JRE. Configure JAVA_HOME using, for example, the following command:
   ```
   ADDENVVAR ENVVAR(JAVA_HOME) VALUE('/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32-bit') LEVEL(*SYS)
   ```
2. Install the following PTF: SI33783 - JVA-RUN JDK60-32 Improved exception handling.
3. Verify that PTF: SI32095 - JVA-RUN JDK60-32 UnsatisfiedLinkError calling RPG native method is installed on your system. It may be marked as SUPERCEDED.
4. To verify the active Java version, open a QSH command and type:
   ```
   java -version
   ```
   Make sure the following tokens are present in the response: "1.6.0", “IBM J9 VM” and “ppc-32”.

Note: The installation was tested on version:
```
Java version "1.6.0"
Java(TM) SE Runtime Environment (build pap3260sr3-20081105_04(SR3))
IBM J9 VM (build 2.4, J2RE 1.6.0 IBM J9 2.4 OS400 ppc-32 jvmap3260sr3-20081105 (JIT enabled, AOT enabled)
```

Preparing for the runtime installation

In general, the process of runtime installation on an OS400 V6R1 does not differ from the process used for older versions of OS400 and previous versions of JIS. Refer to the JIS product documentation for instructions.

Note the following:

- When creating the runtime installation using the “Create Runtime Installation” wizard, it is no longer necessary to select the “Windows” platform when creating a runtime installation only for OS400
- When creating the runtime installation only for OS400, the Windows JRE is no longer part of the runtime installation
- The supported option for installation is using “shared folders” not “ftp”
- Before running the `JSINSTALL` command on the AS400 make sure the user can work with directory entries. Use the `WRKDIRE` command to define this
- The embedded Jetty web server is now enabled by default, and can be used as the primary web server for downloading the application resources

Retirement of Product Components

The following product features are no longer supported:

- The context sensitive help in ACE has been disabled
- The following classes have been removed from the product installation:
  - Analog Clock utility: AnalogClock.class, AnalogClockDisplay.class
  - Calculator utility: CalcButton.class, PocketCalc.class
  - Other utilities: Animation.class, ImageWindow.class, TestJVM.class

  We recommend customers still using these utilities to switch to other commercial or open source alternatives.

New Features Included in JIS 9.0.4

The following new features have been added for JIS 9.0.4

- Logging Improvements
- Simplifying JIS Windows Service Configuration
- Using JAM as an Applet in JIS Standalone Server
- JIS Administrator Command Line Operations
- Modifications made to the J2EE Deployment Procedure (XHTML only)
- Upgrade to Jetty 6.1
- Running the JacadaProxyServlet as part of the JIS Server
- Reduction of the size of the XHTML file
- Allowing the User to Adjust the Java Client Debug Level
- Post Class Path
- New Methods for Handling User Variables
- Rebranding

Logging Improvements

Logging functionality has been improved:

- The architecture parameter 32bit or 64bit is now written to the log
- The current time zone is now written to the log
- The time stamp written to the log now includes milliseconds
- The AM/PM marker has been removed and replaced with 24 hours time

Simplifying JIS Windows Service Configuration

Deploying JIS as a Windows service has been simplified. When running the `JBSToService.exe` utility after creating a JIS runtime installation folder for Windows (do not run `JBSToService.exe` from the JIS installation folder itself), most of the default values are calculated correctly as follows:

1. The code is able to automatically locate `JBSService.exe`
2. The code cleans up and uses the command line from the existing `jacadasv.bat` when launching the service (there's no longer a need to clean up the % signs)
3. The default ini settings are read from the `jacadasv.ini` of the runtime installation (note that by default there's no need to specify the settings)
4. The service log is automatically created in `..\classes\logs\JBSService.log`

In addition, the log messages have been improved and time stamps have been added.

Backward compatibility

The new implementation maintains backward compatibility with existing `JBSToService` command line options. For example for an application named XHTMLV9:

1. Create the service:

```
C:\XHTMLV9\bin>JBSToService.exe -c
Service name: JISSvc
Display name: JIS Service
Description: Controls the running of a JIS Server
Path to executable: C:\XHTMLV9\bin\JBSService.exe
```

2. Service “JIS Service” now appears in the services control panel. You can start and stop it using the standard services panel.
3. Remove the service:

```
C:\XHTMLV9\bin>JBSToService.exe -r
```

Using JAM as an Applet in JIS Standalone Server

JAM can now run as an Applet also when using the standalone server. The main advantage of this configuration is that it does not require opening any ports in the Firewall. In previous versions when running the JIS server on Unix, users had to either use an X-terminal for running JAM or open several ports in the Firewall in order to run JAM from a Windows workstation. This is no longer necessary.

To access the JAM Applet from the development environment use the following URL: http://localhost:8080/jisadminservlet. In production configuration replace `localhost:8080` with your server address and port.

When running as an Applet, JAM is password protected. The default username/password is: jisadmin/jisadmin.

Secure Login to JISAdminServlet

When JAM is running as an applet, the login to JAM is secured and requires a username and password (required when accessing `http://<host>:<port>/JISAdminServlet)`. The username and password can be specified in the `jacadasv.ini` file, under the `[HTTP]` section, using the JAMUsername and JAMPassword keywords. The value of the JAMPassword can be written as an encrypted password. Generate the encrypted password using the batch file located in `<JIS installation folder>\JacadaFiles\utils\web\jetty\encodePassword.bat`. If the JAMUsername and JAMPassword are not specified in the ini file, jisadmin is used for both the username and password.

> **Note:** When accessing JAM via Internet Explorer, you are required to enter your user name and password twice

Example 30. JIS 9.0.4 Generating An Encrypted Password

Example for generating an encrypted password:

1. 1. From a command prompt, execute:

```
C:\XHTMLV9\utils\web\jetty>encodePassword.bat mypass
…
OBF:1xfd1zt11uha1ugg1zsp1xfp
MD5:a029d0df84eb5549c641e04a9ef389e5
```

2. 2. Add the following setting to `jacadasv.ini`:

```
[HTTP]
JAMUsername=myuser
JAMPassword=OBF:1xfd1zt11uha1ugg1zsp1xfp
```

JIS Administrator Command Line Operations

The standalone version of JAM now provides command line interface for performing operations such as shutting down the server, suspending connections of new users, resuming activity on the server and checking the status of the server.

To use the command line interface, open a command prompt, navigate to the `<JISRoot>` folder and issue a `JAM –x <command>` as shown below.

```
jam -x  shutdown <time in minutes>
```

Closes a JIS server after a time interval specified in minutes (when the time interval is not specified, the server is closed immediately).

```
jam -x  suspend
```

Suspends connections of new users to the JIS server.

```
jam -x  resume
```

Resumes activity on the JIS server.

```
jam -x  status
```

Checks if the JIS server is running.

The “status” command has the following return codes:

Table 12 - 52 : JIS 9.0.4 JAM Status Command Return Codes

Code	Description
1	The server is running
-1	The server is not running or there is a communication problem between JAM and the server.

In order to check the value of the status command, you can create the following CheckServerStatus.bat file in your `<JISRoot>` folder:

```
@echo off
call jam -x status
IF %ERRORLEVEL% EQU -1 goto servererror
echo CheckServerStatus: server is Ok
goto exit

:servererror
echo CheckServerStatus: something is wrong with the server or with the
connection from jam to the server
```

```
:exit
```

Modifications made to the J2EE Deployment Procedure (XHTML only)

The JIS common installation for J2EE will no longer attempt to update the classpath of an application server or deploy the application EAR files automatically.

After installing the common installation and before deploying the application ear file, add the jar files placed in the common installation `\lib` folder into the application server's classpath.

As of JIS 9.0.4 the jar files are:

- jacadasv.jar
- Tidy.jar
- sac.jar
- cssparser-0.9.5.jar

Adding jar files to an application server's classpath is an application server specific procedure. Please consult your application server documentation. Specifically, adding jar files to the WebSphere application server is documented in the XHTML user guide.

Upgrade to Jetty 6.1

The embedded Jetty web server bundled with JIS has been upgraded to version 6.1.19. In addition, a few more configurations have been introduced:

1. The ability to use HTTPS only and disable the HTTP port. Use the following `jacadasv.ini` setting:

   ```
   [HTTP] SupportHttpsOnly=1
   ```

2. The ability to disable directory browsing. Directory browsing is enabled by default. Use the following `jacadasv.ini` setting to disable it:

   ```
   [HTTP] AllowDirectoryBrowsing=0
   ```

3. The ability to hide some of the server resources from the client.

   ```
   [HTTP] ProtectedResources=/classes/MyFile1.txt,/classes/MyClass.class
   ```

   When trying to access these resources from a URL such as: http://myserver:8080/classes/myfile1.txt the client will receive a 404 response. The following resources are protected by default:

   ```
   /classes/http.xml
   /classes/jetty-jmx.xml
   ```

```
/classes/jacadasv.ini
/classes/jacadasv.policy
/classes/jcedit.res
/classes/jrodefaults.ini
/classes/license.dat
/classes/proxyConfiguration.xml
/classes/proxyHttp.xml
/classes/ServerConfiguration.dtd
/classes/ServerConfiguration.xml
/classes/JettyKeyStore
/classes/cst/jacadasv.jar
```

4 The `ResourceBase` property now defaults to the `RtRootDir` property.

Backward compatibility

In this release action definition names specified in `ServerConfiguration.xml` such as "`Xhtml`" or "`FreeSession`" are case sensitive (they were case insensitive until now). JIS recognizes action names as they are written in `ServerConfiguration.xml`, as all upper case or as all lower case. For example, the action `/FreeSession` can also be used as `/FREESESSION` or `/freesession`.

Running the JacadaProxyServlet as part of the JIS Server

The Java client can now communicate directly with the JIS server using HTTP/S without having to deploy the JacadaProxyServlet to an external Servlet engine. The Servlet is now run using the embedded Jetty 6.1 servlet engine.

To configure a client to connect to the server using HTTP, add the following Applet parameters:

```
<PARAM name = "UseHttp" value = "true">
<PARAM name = "UsePorts" value = "false">
```

In addition, a new .html page (`<ApplName>-JavaClientHttp.html`) is generated during Generate Runtime. The page contains the necessary definitions for the Java client to connect to the server from which it was downloaded via HTTP/S.

The embedded ProxyServlet always works in non persistent mode. The request used for sending messages from the server to the client is closed by the server and opened by the client after specific protocol messages. This ensures that the client does not keep an open connection to the server for long periods of time.

Configuration When using the embedded ProxyServlet, the following `jacadasv.ini` settings replace settings which were configurable in the `web.xml` when deploying the ProxyServlet as a standalone component:

```
[JISProxyServlet]
```

`HideException` - hide exceptions thrown by the ProxyServlet from the client [default: 0].

`EnableTestServlets` - enables the test servlets for researching communication problems [default: 0]

`GetClientIPFromHttpHeader` - allows to retrieve the client IP from an HTTP header [default: 0]

`ClientIPHttpHeaderName` - the name of the HTTP header from which to read the client address [default is empty]

Logging The embedded ProxyServlet writes log messages to the standard server log. There are no longer jac-<sessionid>.log files.

Backward compatibility You can still package the standalone JacadaProxyServlet classes and deploy them to your desired servlet engine. However it is recommended to start planning their migration to the embedded proxy servlet.

The HTTP communication mode is optional. You can still use the standard ports communication.

The HttpDebugLevel applet parameter is now obsolete.

Reduction of the size of the XHTML file

This feature reduces the HTML page size generated by JIS and in this way reduces the network bandwidth consumed by JIS applications. In addition it also accelerates the generation of the XHTML page in runtime:

- Position related style attributes of HTML elements are no longer part of the page itself, instead they are externalized into a CSS. The CSS is generated in runtime the first time the sub application window is accessed.
- The generated CSS for a sub-application is sent to the browser once per session.

We predict that this will reduce the page size generated by JIS by approximately 30%.

The feature is enabled by default and can be disabled by setting the following parameter to 0.

```
[XHTML]
OptimizeStyleAttributes=0 (1 is the default value)
```

Allowing the User to Adjust the Java Client Debug Level

It is now possible for the end user to set the debug level of the Java client logs in the current session. This can be done by selecting Application>File>Adjust Debug Level or by clicking on a key combination (defined by the JIS developer). The key combination can be defined in the `DebugLevelAdjustKey` (set to any valid single character) and `DebugLevelAdjustKeyModifier` (set to Ctrl, alt or Shift) applet parameters inside the HTML page. When not specified, the default key combination is ALT+d.

This feature is useful for debugging resource problems in the Java client. The user can start the session with debug level 1 and only increase the debug level when a problem such as slowdown is observed.

Post Class Path

This allows running the JIS server with an additional set of jars. It is possible to add an extra token to the classpath, which is appended at the end of the default JIS classpath. This is done in the `jacadasv.ini` file, where you can set, within the `[VMCommandLine]` section, the `PostClasspath` setting to list all the required jar files, which are not part of the JIS default classpath.

Example 31. JIS 9.0.4 PostClasspath Setting

```
[VMCommandLine]
PostClasspath=c:\jdbc\jdbc.jar;myapp.jar
```

New Methods for Handling User Variables

The following DoMethods were introduced:

```
DoMethod: Receiver: System Method: logSharedUserVariables Parms: ( <debug
level> )
DoMethod: Receiver: System Method: logUserVariables Parms: ( <debug level> )
```

The following public APIs were introduced:

```
/**
Retrieve all user variables
@return Map of variables in key,value pair format.
*/
public Map getUserVariables();
```

```
/**
print to the server log file all user variables
@param debugLevel variables will be printed when the server log debug level
is equal or higher than debugLevel
*/
public void logUserVariables(int debugLevel);

/**
Retrieve all shared user variables
@return Map of shared variables in key,value pair format.
*/
public Map getSharedUserVariables();

/**
print to the server log file all shared user variables
@param debugLevel variables will be printed when the server log debug level
is equal or higher than debugLevel
*/
public void logSharedUserVariables(int debugLevel);
```

Example 32. JIS 9.0.4 Handling User Variables From Within an ACE Method

```
Action: Enter
Trigger: 18000 WaitIndicator: True ScrambleName: False MoveMode: MoveNone
Description: This method presses the Enter key on the host, and then
proceeds to the next screen.
Update: Fields: ( _All_Fields_ ) From: TheWindow To: TheScreen
DoMethod: Receiver: System Method: logSharedUserVariables Parms: ( 50 )
DoMethod: Receiver: System Method: logUserVariables Parms: ( 70 )
DoMethod: Receiver: SubApplication Method:
SetCursorPosOnScreenAccordingToFocusedControl Parms: ( )
HostType: AidKey: AidEnter RemainInScreen: False
DoMethod: Receiver: SubApplication Method: MoveAccordingToHost Parms: ( )
```

Example 33. JIS 9.0.4 Handling User Variables From Within a Server Side Extension

```
package appls.TESTB48.server.user;

import cst.server.general.*;
```

```
import java.util.*;
public class GeneralSubApplication extends
appls.TESTB48.server.original.GeneralInternalSubAppl {
   public GeneralSubApplication (Globals globals_parm){
      super (globals_parm);
      return ;
   }
   public void u_Enter(int lParam) {
      super.u_Enter(lParam);
      Map vars = globals.system().getSharedUserVariables();
      if (vars != null) {
         Set keys = vars.keySet();
         Iterator iterator = keys.iterator();
         while (iterator.hasNext()) {
            String key = (String)iterator.next();
            String value = (String)vars.get(key);
            // do something useful
         }
      }
   }
}
```

Rebranding

The software and product documentation has been rebranded to suit Software GmbH standards.

Detailed Description of Fixes Included in JIS 9.0.4

Server

P#312892 By default when an exception occurs in an ACE method triggered from a user action such as "button click", the exception is caught, logged and ignored without terminating the session. The drawback of this approach is that problems are not reported to the user and might be left unnoticed.

Now, you can change this behavior so that the method exception will terminate the session and generate a session dump and error page using the following `<AppName>.ini` setting:

```
[Program] TerminateMethodOnException=1
```

(#133)

P#312833: Exception when trying to read messages from a non-existent screen. (#127)

P#313748 Setting a default value for a combo box, sometimes terminated the session. (#135)

P#316124 When running the JIS scalable server with multiple processes, sessions running on different processes sometimes had the same session ID. (#153)

P#317794 When performing complex navigations an exception occurred and the session was terminated. (#157)

Java client

P#318634 Fixed regression problem related to the usage of: `getStarter().activate(..., true).`(#174)

P#306728: The ToolTips were not always properly displayed. (#134)

P#313342 Two new APIs have been added to JacadaStarter:

- `public void printGui (boolean wait, int timeout)` – prints the window and waits for the specified timeout before returning control to the caller.
- `public void printHostScreen (boolean wait, int timeout)` - prints the host screen and waits for the specified timeout before returning control to the caller. These APIs can be used from a Java extension to prevent a locked printer from locking the session. (#139)

P#309576 Flicker problem related to keyboard buffering has been fixed. (#114)

P#308001 When using the printer emulation for printing in landscape, the margin settings provided by the server were not interpreted correctly by the printer dialog box. In addition, changing the margins via the printer dialog box did not work correctly. (#152)

XHTML Client

P#312379: When using template extensions, JIS loads the template HTML, cleans it up and converts it to a DOM object using a 3rd-party component called Tidy, then merges the template with the generated page. This process occurs

once, the first time a given subapplication is loaded, and is then cached until the server is restarted. When the provided HTML did not include a proper closing `</SCRIPT>` tag, the merged page did not function properly. (#126)

P#314322: In applications deployed on the standalone server and including Popup Support, an error page wasn't displayed when an error forced the session to close. (#148)

P#316850 When using the redirection proxy, the HTTP version of the request was sometimes parsed incorrectly. (#154)

P#317532 It was not possible to change the colors of radio buttons within a group box component using Skins. (#158)

ACE

SAGSIS P#310851 When using user profiles and placing a file without an extension in the ..\users\<myuser> P# 309553 folder, JAM was unable to connect to the server. This has been fixed. (#119)

P#309553 On a Chinese machine, in the Host screen image, Japanese characters were displayed instead of Chinese. Now, to display Chinese characters, set `specific.ini [Program] Host DBCS Font Character Set=134`. (#116)

P#311825 `BuildApp.dtd` has been added to the installation. `BuildApp.dtd` is a description file for the `BuildApp.xml` file used when running ACE from the command line. (#143)

P#317131 The `getCurrentText()` method did not function properly when applied to a combo box component. It is now possible to retrieve the combo box values as a semicolon separated string. (#155)

P#315143 The Java compiler command in Options->Runtime Generations options in ACE was limited to 512 characters. As one can specify a long list of source locations in this field and exceed this limit, the limitation has been increased to 1024 characters. (#144)

Limitations of JIS 9.0.4

The following are known limitations of JIS 9.0.4:

- When upgrading a previous version of JIS, some icons are not updated to the new rebranded icons

- When using IE8 to run a JIS XHTML application which is deployed to an application server, it is not possible to open more than one JIS session from the same browser window
- When starting a session in IE8, the browser toolbar is sometimes shown in black
- When starting a JIS XHTML application, deployed to WebSphere 7, an error is printed to System.out. To avoid this error, first copy the JIS jar files from `<JISCommon>\lib to <WASRoot>\AppServer\lib\ext` and then add the following Java system property:
  `JacadaCommonDirectory=<Installation directory of JISCommon>`
- It is not possible to run the JIS server using a 64-bit JRE. Use a 32-bit JRE instead
- The JIS common installation for J2EE deployment cannot be installed on a Windows 2008 64-bit machine
- When running JIS XHTML using Firefox 3.5, the calendar window includes the source URL at the top of the window, in IE the URL does not appear
- Though the product has been rebranded, the term "Jacada" still appears in several product entities such as directory names, urls, javadoc, method names, class names, etc
- When the user updates an existing runtime installation for Unix to the same location on the Windows machine, and changes the Unix installation folder or JVM folder using the installation wizard, the new folders are not updated into the `jacadasv.ini` file
- XHTML client: The `SupportHttpsOnly` setting cannot be used when running the standalone server with a RedirectionProxy since the RedirectionProxy uses HTTP for internal server communication messages
- When running the JIS server as a Windows service, when stopping the JIS server from the administrator utility, the service is still displayed as 'started' in the Windows services panel. Stop the service from the services panel to clear out this inconsistency
- When running Jam on WebSphere with clustering, not all the clustered servers are displayed
- When running a JIS XHTML application deployed to WebSphere 6.1 under a high load, WebSphere 6.1 may crash with the following error:

  ```
  1XHEXCPMODULE Compiling method: cst/server/applicat/convertr/runtime/
  RunTimeApplication.moveToNextFullScr(Lcst/server/applicat/SubApplication;Lcst/
  server/api/ScrId;I)Lcst/server/applicat/SubApplication;
  ```

  To resolve this, exclude the `moveToNextFullScr` method from the WebSphere JIT compiler by adding the following parameter to the Java command line used for running WebSphere:

  ```
  -Xjit:exclude={cst/server/applicat/convertr/runtime/
  RunTimeApplication.moveToNextFullScr (Lcst/server/applicat/SubApplication;Lcst/
  server/api/ScrId;I) Lcst/server/applicat/SubApplication}
  ```

- When running JIS XHTML application deployed to WebSphere 6.1 under a high load, WebContainer threads may become unusable after the following exception is logged:

  ```
  java.lang.NullPointerException
  at com.ibm.io.async.AsyncFuture.completed(AsyncFuture.java:111)
  at com.ibm.io.async.ResultHandler.complete(ResultHandler.java:195)
  at com.ibm.io.async.ResultHandler.runEventProcessingLoop(ResultHandler.java:743)
  at com.ibm.io.async.ResultHandler$2.run(ResultHandler.java:873)
  at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1473)
  ```

  To resolve this, install the following IBM Java update: http://www-01.ibm.com/support/docview.wss?uid=swg1pk72336

JIS Interface Server 9.0.3 Release Notes

Installation & Upgrade Information

Supported Platforms

JIS Interface Server 9.0.3 is supported on the following platforms:

- Windows 2003 Enterprise Edition
- Solaris 9 & 10
- AIX 5.3
- AS/400 V5R2

Note: When a vendor no longer supports an OS version, Software GmbH will discontinue support for this OS version effective immediately.

Recommended Configurations

Software GmbH provides support for Java versions, browser versions and application server versions supported by their respective vendors. Being so, when a vendor no longer supports a Java version, browser version or application server version, Software GmbH will discontinue supporting that version as of the next JIS service pack level delivered by Software GmbH. Although it may be technically possible to run a new version of JIS using an unsupported version, Software GmbH cannot continue to support configurations that are no longer supported by their vendor.

ACE

The ACE interactive development kit has been tested on the following operating systems:

- Windows XP Professional
- Windows Vista

Clients

The Java Client has been tested on the following operating systems, browser and Java versions:

- Windows XP Professional
- Windows Vista

Table 13 - 53 : JIS 9.0.3 Java Client Supported Browser and Java Versions

Browser	JRE
IE 7	Sun JRE 1.6.0_07
Firefox 3.0	Sun JRE 1.6.0_07 (partially tested)

The XHTML client has been tested with the following operating system and browser versions:

Table 13 - 54 : JIS 9.0.3 XHTML Client Supported Operating Systems and Browsers

Operating Syste	Browser
Windows XP Professional	IE 6, IE 7 & Firefox 3.0 (partially tested)
Windows Vista	IE 6, IE 7 & Firefox 3.0 (partially tested)

Standalone Server

The JIS standalone server has been tested in the following environments:

Table 13 - 55 : JIS 9.0.3 Standalone Server Supported Environments

Operating System	Java Version
Windows 2003 Enterprise Edition	Sun 1.4.2_10 & Sun 1.5.0_09
Solaris 9 & 10	Sun 1.5.0_09
AIX 5.3	IBM 1.4.2
AS400 V5R2	IBM 1.4.2

J2EE Deployment

The runtime of the JIS XHTML client has been tested for deployment in the following environments:

Table 13 - 56 : JIS 9.0.3 XHTML Client Supported Runtime Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WebLogic 8.1 SP6	JRockit 1.4.2	Windows 2003 Enterprise Edition
WebSphere 6.1.0_17	IBM JDK 1.5	Windows 2003 Enterprise Edition
WebSphere 6.1.0_17	IBM JDK 1.5	Solaris 10
WebSphere 6.1.0_17	IBM JDK 1.5	Red Hat Linux AS4
Tomcat 4.1.29	Sun JRE 1.4.2_06	Windows 2003 Enterprise Edition (partially tested)

JIS Java Proxy Servlet

The runtime of the JIS Java proxy servlet has been tested for deployment in the following environments.

Table 13 - 57 : JIS 9.0.3 Java Proxy Servlet Supported Environments

Application Server or Web Container	Java Runtime Environment	Operating System
WebSphere 6.1.0_17	IBM JDK 1.5	Windows 2003 Enterprise Edition

AS400 Components

The DDS compiler has been tested on the following operating systems:

- OS400 V5R3
- OS400 V6R1 (partially tested)

Installation & Upgrade Instructions

JIS 9.0.3 is the next service pack of the JIS 9.0 major release. Unlike previous JIS 9.0 service packs (PTFs) which were delivered as a differential installation and had to be installed on top of the JIS 9.0 main release, JIS 9.0.3 is delivered as a standalone installation.

JIS 9.0.3 can be installed as follows:

- On a clean machine which does not contain a previous installation of JIS. This installation requires entering your existing product CDKey
- As an upgrade to an existing JIS 9.0 or JIS 9.0 PTF or service pack installation such as JIS 9.0A07 or JIS 9.0B
- Side by side with an existing JIS 9.0 installation on the same machine. This installation requires entering your existing product CDKey
- The JIS 9.0.3 installation no longer installs the following 3rd-party components:
  - Acrobat reader – can be downloaded from the Adobe web site
  - Microsoft JVM – is no longer supported by Microsoft
  - Aladdin Hasp drivers – are no longer required for running JIS
  - Wise demo version – can be downloaded from the Symantec web site

- The JIS 9.0.3 installation now automatically installs the following components without requesting user confirmation:
  - Product documentation
  - Release notes document
  - Product tutorials

Note: When upgrading an existing JIS version to JIS 9.0.3, the uninstall procedure for JIS 9.0.3 will only remove components installed by JIS 9.0.3.

AS400 Components Installation

In JIS 9.0.3, the AS400 components are no longer installed as part of the main product installation. Instead, a separate AS400 components installation is supplied under the "`..\utilities\Jacada iSeries Components`" folder and can be installed as a standalone component. The AS400 components installation will now prompt for entering the product CDKey in order to install the AS400 specific development tools.

JIS Common Installation for J2EE Deployment

JIS 9.0.3 includes an update to the JIS Common installation component. Customers using XHTML J2EE deployment will need to upgrade their common components installation to the version delivered with JIS 9.0.3. The JIS common components' installation will no longer add the JIS specific jar files to the application server's classpath. This operation has to be done manually for each application server. If you intend to deploy JIS to WebSphere 6.1 refer to the Jacada_for_XHTML.pdf page 632 section "Add the Jacada JAR Files to WebSphere's Classpath".

JIS Runtime Installation for Proprietary Server

Since the JIS installation no longer includes the Wise demo version, customers will have to download the Wise Installation Studio from the Symantec web site. JIS 9.0.3 was tested using Wise Installation Studio 7.0 SP1.

The JIS runtime installation now adds the ability to automatically FTP and configure the JIS proprietary server on supported UNIX platforms. See an updated procedure in the JIS release notes.

License

JIS 9.0.3 does not require customers using V9.0 and above to obtain a new CDKey or a new runtime license.

Customers upgrading from V8.1 or earlier will need to obtain a new runtime license which replaces the now obsolete RTCP key. Without a valid runtime license, JIS is limited to running 10 concurrent sessions.

Usage of Dongle

JIS 9.0.3 no longer requires the use of the Aladdin Hasp dongle.

New Features Included in JIS 9.0.3

The following new features have been added for JIS 9.0.3:

- Changes in the Product Name
- Runtime Installation Improvements
- Simplifying the Printing Emulation Configuration (XHTML)
- Improved Host Language Support
- Java Client "About" Dialog Box
- “Host Print Transform" Printing using Java Services
- Exposing the XHTML Page DOM for Java Extensions
- Enforcing Strong Encryption
- AutoSkip Supported in XHTML

Changes in the Product Name

As a result of the acquisition of the Jacada application modernization product line by Software GmbH, we have begun the process of updating the product name to suit the company standards. The product name is now JIS, and we have begun to implement this throughout the product. This has not yet been implemented in the tutorials and in the documentation.

Runtime Installation Improvements

The runtime installation process now enables installing and deploying JIS automatically on UNIX platforms. At the end of the Wise installation process, an ANT script will be invoked to transfer the files to the UNIX machine and

configure the JIS server. Two optional, new dialog boxes in the installation process enable implementing this feature. Refer to the runtime installation process in the documentation for details.

UNIX Machine Prerequisites:

- FTP access enabled
- Telnet access enabled
- Unzip command installed

Changes in the installation process

In the following screen, when selecting to automatically transfer the installation to UNIX by FTP, you will be required to enter the IP address or name of the UNIX machine where the JIS server is to be installed, the name and password of a user who has permissions on the machine to connect using FTP and Telnet and the postfix of the shell command prompt string to be used by the UNIX machine.

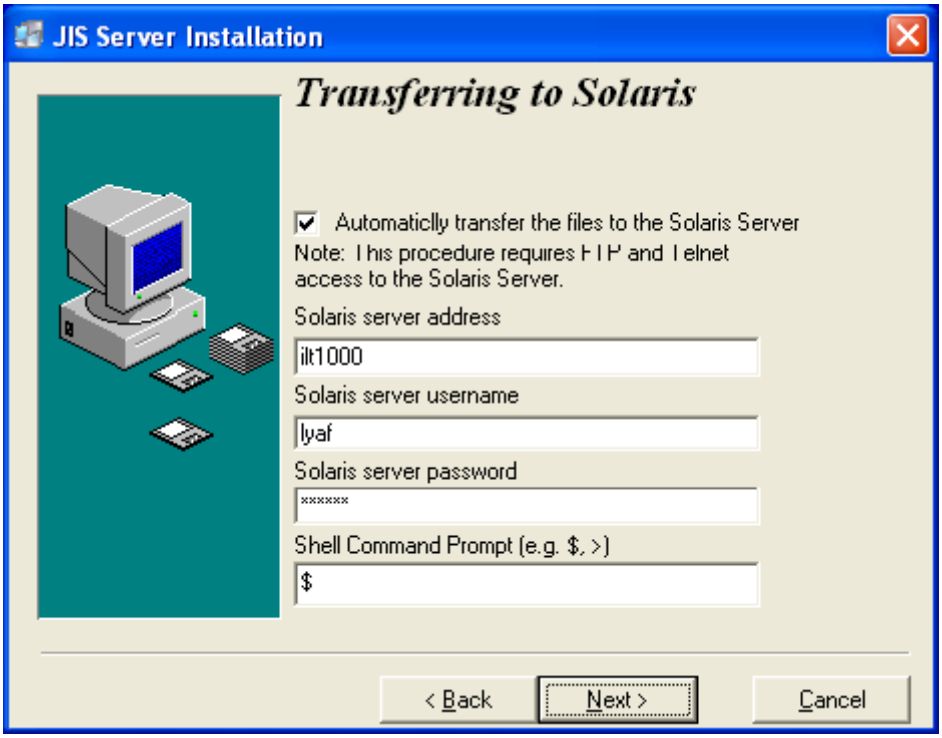

Figure 22. JIS 9.0.3 Transferring to Solaris

The screen which follows the above screen is the Application Configuration screen, where you are required to select whether to replace the configuration files or to preserve existing configuration files.

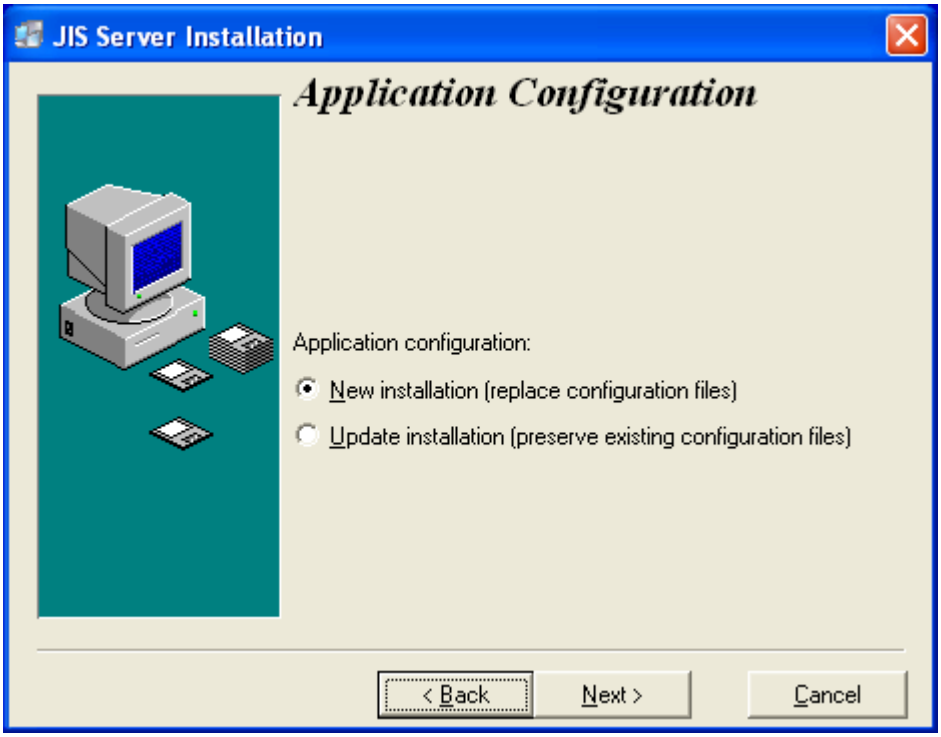

Figure 23. JIS 9.0.3 Application Configuration

> **Note:** The new installation procedure is only available for Solaris, AIX and Linux.

Simplifying the Printing Emulation Configuration (XHTML)

The printing emulation configuration for JIS XHTML has now been simplified, and default values are provided for most parameters.

The following parameters now have defaults which are suitable for most configurations and no longer need to be defined in the `<ApplicationName>.ini`:

```
[TN5250 Printer]
WorkRootURL
WorkRootDirectory
SpoolDirectory
XSLTforXMLtoHTML
[Printing Handlers] section
```

The following parameters still need to be configured in order to enable printing:

```
[GUISys TN5250]
```

```
; enable host printing
Printer=1
[TN5250 Printer]
; set the device name
LUName=<Device name>
```

Improved Host Language Support

Introduction

JIS support of host languages has been simplified. JIS has now integrated the "descriptor" mechanism, which enables support of more than one host language on the same server requiring minimal configuration. Users no longer need to use the complex and error prone LanguageDescriptorFactory extensions. This version of JIS also clarifies the level of support to languages that were not previously supported.

An additional language related enhancement which has been added supports printing special characters to the log file, improving debugging capabilities.

> **Note:** Language Descriptors can still be customized at the project level to maintain backward compatibility.

Using this Feature

The following details how to enable and configure this feature on the server, the Java Client and the XHTML client.

Server

Add the following `<ApplicationName>.ini` setting to enable the feature:

```
[Emulator]
LanguageDescriptorEnabled=1
```

> **Note:** Applications that have used LanguageDescriptorFactory extensions in previous versions will also need to add this setting when upgrading.

Java Client

Configuring host language support for Java clients:

1. Specify the host language used by the client using the following Applet parameter: `<PARAM name = "LanguageDescriptor" value = "<Language Name>">`.
2. Make sure the client operating system supports the language specified in its "Regional Settings" and that the language specific fonts, if there are any, are installed.
3. When using Chinese, Japanese, Korean or Thai, add the clcharsets.jar file to the ARCHIVE tag and the clcharsets.cab file to the Cabbase parameter in the launcher HTML page. If you are using one of these languages and the clcharsets archive is not added this will cause the LanguageNotSupported.html page to be presented when starting a session.
4. When using some languages (such as Japanese, Chinese or Thai) the default fonts used by the JIS host screen and the JIS JITGUI sub-application do not display the screen contents correctly. Instead you may see square signs or question marks. To fix this experiment with the following Applet parameters:
   `<PARAM name = "CourierFontType" value = "Courier">` to control the font in the JITGUI and dynamic areas.
   `<PARAM name = "EmulatorFontName" value = "Courier">` to control the font used by the Host Screen.

XHTML Client

Configuring host language support for XHTML clients:

1. Specify the Language used by the client using the following URL parameter or post data or HTTP header "`LanguageDescriptor=<Language Name>`". Note that you cannot pass this parameter to the `<AppName>-xhtml.html`. You either have to code it in the `<AppName>-xhtml.html` itself or write the full URL in the browser's address bar such as `/XHTML?JacadaApp`.
2. Make sure the client operating system supports the language specified in its "Regional Settings" and that the language specific fonts, if any, are installed.

Customization

It is still possible to customize the LanguageDescriptor provided by the product for project specific requirements. In order to do this, the language descriptor classes must be named as follows:

Client side

```
appls.<AppName>.user.User<LanguageName>LanguageDescriptor
```

Server side

```
appls.<AppName>.server.user.User<LanguageName>LanguageDescriptor
```

Log Files

The server log and Java client log now displays field content encoded using the current session language encoding. To view the log files with the correct encoding, we recommend viewing the log file from a client machine which supports the encoding used by the sessions and use an encoding aware text editor such as Wordpad.

Backwards Compatibilility

Applications which do not use LanguageDescriptors

Existing applications that use the application level language setting in ACE, and do not use language descriptors, will continue working as before.

Applications which use LanguageDescriptor extensions

Existing applications already using language Descriptors implemented as extensions should first try to use the internal descriptors and remove and discontinue the usage of the extensions.

If you must continue using the extensions, then set the following:

In the `<ApplicationName>.ini` file:

```
[Emulator]
LanguageDescriptorEnabled=1
LanguageDescriptorFactory=appls.<AppName>.server.user.
ILanguageDescriptorFactory
```

In the HTML Launcher:

```
<PARAM name = "LanguageDescriptorFactory" value = "
appls.<AppName>.user.ILanguageDescriptorFactory">
```

Where `<AppName>` is the name of the JIS application.

```
<PARAM name = "LanguageDescriptor" value = "<Language name from the
Supported Languages table">
```

In addition, if you are using an existing ChineseLanguageDescriptor class on the server and client, you'll need to:

1 Change the `isLanguageSupported ()` method as follows:

```
public boolean isLanguageSupported(String language) {
```

```
        return language.equalsIgnoreCase("Chinese");
    }
```

2 In the `ILanguageDescriptorFactory` class, replace occurrences of "Chinese (Simplified)" with "Chinese".

Parameters

This section provides a reference to the configurable parameters:

LanguageDescriptorEnabled

Enables the "Descriptor" mechanism within JIS. Once this parameter is set, both the internal language descriptors and language descriptor extensions are enabled.

Configuration file: `<ApplicationName>.ini`.

Section: `[Emulator]`

Possible values: 0, 1 (default - 0).

Example: LanguageDescriptorEnabled=1

LanguageDescriptorFactory ini setting & Applet parameter

Allows using a server side LanguageDescriptorFactory for backward compatibility. Users upgrading their descriptors from an earlier version, who would like to continue to use the external descriptors, should configure this setting.

Configuration file: `<ApplicationName>.ini`

Section within file: `[Emulator]`

Possible values: Class name (default - uses internal factory.)

Example:

ini setting:

```
LanguageDescriptorFactory=appls.MYAPP.server.user.ILanguageDescriptorFactor
y
```

Applet parameter:

```
<PARAM name ="LanguageDescriptorFactory"
value="appls.MYAPP.user.ILanguageDescriptorFactory">
```

LanguageDescriptor Applet parameter & URL parameter

Name of the language descriptor to be used by this client session.

Configuration file: Java client launcher HTML. Section: Applet parameters

Possible values: Listed in the Supported Languages table.

Example:

Applet parameter:

```
<PARAM name ="LanguageDescriptor" value="Chinese">
```

URL parameter:

```
http://localhost:8080/
Xhtml?JacadaApplicationName=MYAPP&LanguageDescriptor=Chinese
```

Parameters for Backwards Compatible Settings (to be used only when not using the "Descriptor" mechanism)

Conversion File

This value will override the host code page defined by the language descriptor for all sessions.

Configuration file: `<ApplicationName>.ini`. Section: `[GUISys TN5250]`

HostCodePage

This value will override the host code page defined by the language descriptor for all sessions.

Configuration file: `<ApplicationName>.ini`. Section: `[GUISys TN3270]`

RuntimeLanguage

There is no longer a need to specify a specific server side language since the server now supports all languages

Configuration file: `<ApplicationName>.ini`. Section: `[Emulator]`

Supported Languages

The following table lists the languages supported.

Support Level:

- 1 indicates that the language is fully supported.
- 2 indicates that conversion tables exist for this language, but that the language was not tested.

Table 13 - 58 : JIS 9.0.3 Supported Languages

Language Name	Level of Support
EnlishUS	1
Albanian	2
Belorussian	2
Bulgarian	2
Chines	1
Chinese Traditional	2
Croatian	2
Czech	2
Danish	2
EnglishUK	1
French	1
German	1
Greek	2
Hungarian	2
Italian	1
Japanese	1
Korean	1
Macedonian	2

Table 13 - 58 : JIS 9.0.3 Supported Languages

Language Name	Level of Support
Norwegian	2
Polish	2
Portuguese	2
Romanian	2
Russian	2
Thai	1
Serbian	2
Slovak	2
Slovenian	2
Spanish	1
Swedish	2

JIS Language Descriptor to codepage mapping

The following table lists the language descriptor to codepage mappings.

Table 13 - 59 : JIS 9.0.3 Language Descriptor to Codepage Mapping

Language Descriptor	Class Name Prefix	EBCDIC Codepage	ASCII Codepage	Is Double Byte
Default	N/A	Cp037	Cp1252	
Albanian	Albanian	CP80	CP1250	

Table 13 - 59 : JIS 9.0.3 Language Descriptor to Codepage Mapping

Language Descriptor	Class Name Prefix	EBCDIC Codepage	ASCII Codepage	Is Double Byte
Belorussian	Belorussian	Cp1025	Cp1251	
Bulgarian	Bulgarian	Cp1025	Cp1251	
Chinese (Simplified)	Chinese	GB935	GBK	Yes
Chinese (Traditional)	ChineseTraditional	Cp937	Big5	Yes
Croation	Croatian	Cp870	Cp1250	
Czech	Czech	Cp870	Cp1250	
Danish	Danish	Cp1142	Cp1252	
English UK	EnglishUK	Cp1146	Cp1252	
English USA	EnglishUS	Cp1140	Cp1252	
French	French	Cp1147	Cp1252	
German	German	Cp1141	Cp1252	
Greek	Greek	Cp875	Cp1253	
Hungarian	Hungarian	Cp870	Cp1250	
Italian	Italian	Cp1144	Cp1252	
Japanese	Japanese	SJIS	MS932	Yes
Korean	Korean	Cp933	Cp949	Yes

Table 13 - 59 : JIS 9.0.3 Language Descriptor to Codepage Mapping

Language Descriptor	Class Name Prefix	EBCDIC Codepage	ASCII Codepage	Is Double Byte
Macedonian	Macedonian	Cp1025	Cp1251	
Norwegian	Norwegian	Cp1142	Cp1252	
Polish	Polish	Cp870	Cp1250	
Portuguese	Portuguese	Cp037	Cp1252	
Romanian	Romanian	Cp870	Cp1250	
Russian	Russian	Cp1025	Cp1251	
Serbian	Serbian	Cp1025	Cp1251	
Slovak	Slovak	Cp870	Cp1250	
Slovenian	Slovenian	Cp870	Cp1250	
Spanish	Spanish	Cp1145	Cp1252	
Swedish	Swedish	Cp1143	Cp1252	
Swiss-German	Swiss-German	Cp500	Cp1252	
Thai	Thai	Cp838	MS874	
Turkish	Turkish	Cp1026	Cp1254	
Ukranian	Ukranian	Cp1025	Cp1251	

Comments:

- The EBCDIC Codepage is being used by the server when converting information sent and received from the host into an ASCII encoding.

- The ASCII Codepage is being used by the server when converting ASCII encoded bytes into Java characters encoded using Unicode.
- All information sent and received between the clients and the server is encoded using Unicode encoding.
- For extension developers, to obtain the descriptor class name from the “Class Name Prefix” append the string “LanguageDescriptor” to the prefix.
- “Belorussian” also refers to “Belarussian” and “Slovenian” also refers to “Slovene”.

Note: Customers can implement their own descriptors by extending the existing product descriptors.

Example 34. JIS 9.0.3 Extending Product Language Descriptors

```
package appls.IT.server.user;
public class UserItalianLanguageDescriptor extends
ItalianLanguageDescriptor {
   // override here methods from the ILanguageDescriptor interface, for
example:
   public boolean isDBCSLanguage() {
      return super.isDBCSLanguage();
   }
}
```

Recommendations

When using LanguageDescriptors, it is recommended to run the interface server with the default file encoding for the operating system and not use a specific encoding.

JIS clients were tested on a standard Windows XP SP2 operating system version (not a language specific operating system) with the specified languages enabled in the regional settings.

Java Client "About" Dialog Box

The About dialog no longer contains the now obsolete RTCP key information "Serial No:" and "Licensed To:" labels.

"Host Print Transform" Printing using Java Services

When using the AS400 HPT (Host Print Transform) feature, the print job is sent from the AS400 already formatted with all the necessary escape codes required for the print job formatting. In previous releases of JIS the Java client sent this print job into a predefined parallel or serial port on the local PC. The end user had to define port capturing on the local PC for the actual printer.

This feature adds the ability to use the Java print service APIs to implement the same behavior. This way the user does not have to configure the port in advance. Java takes care of this for you. In addition HPT, using the print service, now supports the existing `PrinterEmulationPageOrientation` and `PrinterEmulationPaperType` Applet parameters.

Exposing the XHTML Page DOM for Java Extensions

This feature exposes thepage DOM for project specific extensions from the `OnPageLoad()` event handler of `Appl.java`.

A new `OnPageLoadContext` API:

```
public Document getXhtmlDom()
```

This should be used to retrieve the existing page DOM and change it for project specific requirements.

The DOM retrieved by `getXhtmlDom()` does not reflect style changes made by the existing, project specific, XHTML extensions.

In order to reflect changes made by existing XHTML extensions you need to add the following code to the `onPageLoad()` method:

```
public void onPageLoad(OnPageLoadContext context) {
   // Existing Xhtml extensions
   ...
   // Update the new style settings from the Xhtml extensions into the Page
DOM
   Hashtable styleHash = context.getDataBlock().getStyleHash();
   for(Enumeration e = styleHash.keys();e.hasMoreElements();) {
      Element key = (Element)e.nextElement();
      StyleModifier.updateStyleAttribute(key, (Map)styleHash.get(key));
   }
   ...
   // get the already modified DOM and further manipulate it
   Document xhtmlDom = context.getXhtmlDom(); ...
}
```

Another important note is that while you are modifying the component style inside the DOM object, you have to clone the modified component (see Example 35 below) otherwise the JIS internal code will re-apply the styles set in the code extension and override the DOM manipulations.

Example 35. JIS 9.0.3 Using the XHTML Page DOM in a Java Extension

```
package appls.XHTMLV9.xhtml.user;
import com.jacada.jis.runtime.server.frontend.xhtml.context.*;
import com.jacada.jis.runtime.server.frontend.xhtml.general.XhtmlConstants;
import com.jacada.jis.runtime.server.frontend.xhtml.controls.XhtmlControl;
import org.w3c.dom.*;
import java.util.*;
import com.jacada.jis.runtime.server.frontend.xhtml.controls.Window;
import com.jacada.jis.runtime.server.frontend.xhtml.modifier.StyleModifier;
/**
 * description : Appl.java
 */
public class Appl implements
com.jacada.jis.runtime.server.frontend.xhtml.extension.IUserPageExtension {
    /**
     * Constructor
     */
    public Appl () {

    }

    public void onPageLoad(OnPageLoadContext context) {
        Window window = context.getWindow();
        Vector vControls = window.getAllControls();
        for (int i = 0; i < vControls.size(); i++) {
            // Get the control on the window.
            XhtmlControl curControl = (XhtmlControl)vControls.get(i);
            curControl.setForeground("RED");
            //curControl.setSize(100,10);
            //curControl.setLocation(1,1);
        }

        // Update the new style settings into the Page DOM.
        Hashtable styleHash = context.getDataBlock().getStyleHash();
        for (Enumeration e = styleHash.keys();e.hasMoreElements();) {
            Element key = (Element)e.nextElement();
```

```
            StyleModifier.updateStyleAttribute(key,
(Map)styleHash.get(key));
        }

        // This is an example of manipulating label nodes on a JITGUI
        // screen converted screens have different structure specially
        // if they contain tab controls.
        Document xhtmlDom = context.getXhtmlDom();
        NodeList formList =
xhtmlDom.getElementsByTagName(XhtmlConstants.HTML_FORM);
        Node form = formList.item(0);  // We assume this is the jacadaform.
        but you need to check to make sure
            NodeList formElements = form.getChildNodes();
        for (int i=0; i < formElements.getLength(); i++) {
            Node formNode = formElements.item(i);
            if ("span".equalsIgnoreCase(formNode.getNodeName())) {
                NodeList spanChildNodes = formNode.getChildNodes();
                for (int j =0; j < spanChildNodes.getLength(); j++) {
                    Node spanChildNode = spanChildNodes.item(j);
                   if ("pre".equalsIgnoreCase(spanChildNode.getNodeName()))
{
                        NodeList preChildNodes =
spanChildNode.getChildNodes();
                       for (int k =0; k < preChildNodes.getLength(); k++) {
                           Node preChildNode = preChildNodes.item(k);

if("label".equalsIgnoreCase(preChildNode.getNodeName()))
                           {
                               // Clone the label, change its color to Blue
                               // and update it back to the dom.
                               Element label =
(Element)preChildNode.cloneNode(true);
                               String style = label.getAttribute("style");
                               style = style.replaceFirst("color:RED",
"color:#0000ff");
                               label.setAttribute("style", style);
                               spanChildNode.replaceChild(label,
preChildNode);
                           }
                       }
                   }
                }
            }
```

```
        }
    }

    public void onPageSubmit(OnPageSubmitContext onSubmitContext) {

    }
}
```

Enforcing Strong Encryption

When using XHTML client with HTTPS communication, it is now possible to prevent users from connecting to the server using SSL 2.0 or SSL 3.0, thus enforcing users to connect using the stronger TLS 1.0. It is also possible to enforce users to use strong cipher suites. To configure the supported protocols and cipher suites, two settings should be set in the `jacadasv.ini` file under the `[HTTP]` section: `SupportedProtocols` & `SupportedCipherSuites`.

For example to allow IE7 to use only TLSv1 together with 128 bit cipher suite, add the following settings to the `jacadasv.ini`:

```
[HTTP]
SupportedProtocols=TLSv1
SupportedCipherSuites=SSL_RSA_WITH_RC4_128_MD5
```

These settings are only relevant for XHTML client, when using HTTPS between the client and the server.

AutoSkip Supported in XHTML

AutoSkip is now supported enabling automatically skipping to the next field in the tab order, once the field has been filled and the caret is at the end of this field.

In the `<ApplicationName>.ini` file, in the `[XHTML]` section, configure the `AutoSkipSupport` parameter:

```
AutoSkipSupport=0 does not support using autoskip.
AutoSkipSupport=1 supports using autoskip (default value).
```

Detailed Description of Fixes Included in JIS 9.0.3

Security

DODA-6144 When using the JIS Proxy Servlet, the Http session was not invalidated when closing the Java Client session. (#7)

SAGSIS P#309195 In the XHTML client, JIS error pages were vulnerable to cross site scripting attacks (XSS). This has now been blocked. (#84)

SAGSIS P#309359 The default `http.xml` and `proxyHttp.xml` files have been enhanced, to secure server resources such as jar files and configuration files from being viewed from the clients. (#86)

SAGSIS P#306323 When running the server using the IBM JRE, it is now possible to implement SSL from the server to Mainframe using the internal CSTSSLSocketFactory. (#27)

DODA-6137 Encryption between client and server is now supported when the server runs with IBM JRE. (#9)

Server

DODA-6250 Attempting to change runtime language without creating a language descriptor caused the session to terminate. (#11)

DODA-6408 The server did not correctly support writing Unicode values to the `<AppName>.ini` file. (#17)

SAGSIS P#306730 IND$File upload did not work in V9.0. (#31)

Java client

DODA-6314 As a result of changes in the look and feel of the GUI, radio group components no longer displayed their headers. (#13)

DODA-6353 As a result of changes in the look and feel of the GUI, some button styles created in previous versions were not displayed properly. (#14)

DODA-6385 As a result of changes in the look and feel of the GUI, reordering table columns in ACE sometimes caused the runtime to abort. (#15)

BUG-58500 The Java client property storage now creates one file for each application. (#21)

SAGSIS P#307684 As a result of changes in the look and feel of the GUI, there was a problem with the display of the text of active tabs in applications that did not use themes. (#36)

SAGSIS P#306442 The default value of `UseEventDispatchThread` (which ensures that all UI operations are performed by the AWT event dispatch thread), has been changed to "true". To undo this change set `<PARAM name = "UseEventDispatchThread" value = "false">`, refer to the following backward compatibility issues:

- Code extensions using the `activate()` method may lock the client: Specifically code which calls `getStarter().activate(..., true)` will lock the client. In order to fix this, change the code extensions to call: `getStarter().activate(..., false)`.
- Code extensions which rely on the client threading mechanism may stop functioning.

SAGSIS P#307989 The "List" and "Commands" menus did not work when the menu names were localized. (#35)

XHTML Client

SAGSIS P#306190 Calling the `setSkin()` DoMethod before the list of skins was initialized from the `<AppName>.ini` file, resulted in an exception. (#10)

DODA-6334 When forms which include six digit date fields were submitted, these fields were considered as modified even in cases where they hadn't been changed by the user. (#12)

DODA-6151 When using a combo box from ini, the XHTML client did not trim the ini values before comparing them to the value sent from the client, therefore combo boxes whose values were not changed by the user were reported as modified. (#16)

SAGSIS P#308314 When calling `setText("")` from an XHTML extension for a label component, the rendered HTML label was incorrect, causing components on the HTML page to disappear. (#76)

DODA-6199 When using JIS J2EE deployment in a WebSphere network deployment, the license counter did not update properly and exceptions were printed to the log. (#6)

SAGSIS P#307009 Version 9.0B did not support working with the menu bar extension. This has now been fixed and version 9.0.3 supports working with the menu bar extension. (#33)

SAGSIS P#308972 When submitting a form containing a date control, the unformatted value was displayed briefly. (#85)

ACE

DODA-6408 Single byte Japanese characters were not correctly displayed (#18). A new setting has been added to support this:

```
Specific.ini
[MakeExeParms]
ExpressionType=2
```

DODA-6181 When using menu files which contained simplified Chinese characters, ACE crashed. (#20)

SAGSIS P#307051 The `SetModifiedFlag` and `IsModified` methods did not work. They now work on combo box, check box and radio group receivers as well as the rest of the modifiable controls, except for table controls. (#35)

SAGSIS P#307794 The `HideControl` method is now supported when using the Radio Group receiver. (#45)

SAGSIS P#309076 It is now possible to capture a MOD2 screen from a MOD5 library. (#82)

Innovator

SAGSIS P#304664 After installing Innovator for the first time on a new machine, when trying to create a monitor, an error was displayed. (#26)

Administration Console

It is now possible to activate debug log messages when running the administration console as an applet in J2EE, by setting the Debug Level parameter in the URL. For example: http://localhost:9081/jacadaadmin/admin?debuglevel=70. When a debug level is not assigned the default value is 1. The log messages will be logged to the Java Console of the client browser.

The "Set Debug Level" dialog box, activated from the right-click menu of the session tree, sometimes disappeared from the window frame. (#22)

Emulation

DODA-6136 When an EM control signal appeared within the printer data stream, at exactly the last position in the screen, the printout was corrupted. (#5)

SAGSIS P#306147 From now it's possible to negotiate telnet user variables with the AS400 without specifying an LUName in the `<AppName>.ini`. (#104)

Limitations of JIS 9.0.3

The following are known limitations of JIS 9.0.3:

- The iSeries installation relies on FTP to execute remote OS400 calls. During installation, if one of these remote calls takes more than a minute, the installation will fail.
- In IE6, when using the font family MS Sans Serif, in the CSS definition of the password field, “a” appears instead of the password marker (such as an asterisk). Change the password field to use a font such as Courier New to avoid this problem.
- When running ACE on Windows Vista, the Online help activated from the Help menu (Contents, Cue Cards and Example) does not work. In order to activate the help, download WinHelp from http://www.microsoft.com/downloads/details.aspx?familyid=6ebcfad9-d3f5-4365-8070-334cd175d4bb&displaylang=en
- When running ACE on Windows Vista you must turn off the User Account Control (UAC) feature of Vista in order for ACE to function correctly.
- When using the XHTML automatic server updates and there are popup windows nested on top of each other, closing two or more popup windows using the window closing X quickly, leaves the client locked.
- When upgrading, some icons are not updated to the new rebranded icons.
- When using XHTML deployment to WebSphere, in runtime, error messages are not displayed on the Web page.
- When running the Java client with Java 1.6.0_10, every window in the Applet appears with the triangular exclamation mark icon next to it.
- When the user reinstalls an existing runtime installation for Unix to the same location on the Windows machine, and changes the installation folder or JVM folder using the wizard, the new locations are not updated into the `jacadasv.ini` file.
- When configuring JIS XHTML to use HTTPS, you have to create a valid Java keystore file named JettyKeyStore under the `..\JacadaFiles\classes` folder.

JIS Interface Server 9.0B Release Notes

New Features Included in JIS 9.0B

The following new features have been added for JIS 9.0B:

- Platform Support
- Command-Line Access to ACE
- Improved User Interface
- Additional Enhancements

Command-Line Access to ACE

Command-line access to ACE enables automating the following operations:

- Generate Runtime
- Create Runtime Installation
- Pack/Unpack

The command-line access mechanism allows external tools to run ACE in automatic mode, using an XML file that contains a list of operations to execute in ACE.

The XML file must be named buildapp.xml.

Place the buildapp.xml file in the folder from which you launch ACE (the folder containing the ACE executable).

To launch ACE in automatic mode:

- Use the following command parameter: `-oREMOTE`
  For example, to launch JIS XHTML for 3270 in automatic mode, use the following command line:

```
ACE.EXE -lmp -oREMOTE
```

When running ACE with the -oREMOTE command-line parameter, the buildapp.xml is read and the operations are executed by ACE.

Running ACE in Automatic Mode

When ACE is launched in automatic mode, it creates a GUI. Additional windows, such as the Pack animation window, are also displayed. However, in automatic mode, the ACE GUI is disabled.

When all of the operations run successfully (without errors), no user intervention is required. So, for instance, the Generate Runtime process dialog still opens, but closes automatically when it is done. If, however, there is an error message, such as an alert about missing image files, then this message is shown, and the user needs to click *OK* to close it. For more information, see "Error Handling" on page 301.

Once a valid `buildapp.xml` file has been created and tested, the operations run in ACE without the need for user intervention.

Changes in the Behavior of ACE Operations

The following sections describe changes in the behavior of ACE operations when using automatic mode.

Changes in Generate Runtime

The Generate Runtime process is always carried out for the entire application. It is not possible to specify specific libraries or subapplications. To speed things up, you can use the option to compile only new and modified subapplications.

Changes in Create Runtime Installation with Wise

After creating a runtime installation with Wise, ACE asks whether to launch the newly-created installation. In automatic mode, this question is skipped, and the installation is not launched.

Changes in Pack

When packing an application, it is possible to select the libraries to pack, and to add additional files, but the other steps of the Pack Wizard are not supported. Thus, for example, it is not possible to specify a maximal file size, nor to skip input directories (such as skipping DDS files, installation files and configuration files). All the files are included in the package (including configuration files).

Changes in Unpack

When unpacking an application, existing files are automatically replaced.

The configuration files (the files asked about in the last step of the wizard) are unpacked according to the settings chosen the last time the wizard was run from ACE.

The following example demonstrates various operations, such as opening an application, generating a runtime, creating a runtime installation, and packing and unpacking an application:

Example 36. JIS 9.0B Sample buildapp.xml

```
<Ace>
 <OpenApplication Name="TEST1">
  <GenerateRuntime Type="Java;XHTML" Platform="Windows; Solaris;
                  OS390;AS400;AIX;Linux" NewAndModified="0" />
 </OpenApplication>

 <OpenApplication Name="TEST2">
  <GenerateRuntime Type="Java;XHTML"
           Platform="Windows;Solaris;Linux;AIX;AS400;OS390" />
  <CreateRuntimeInstallation DeploymentType="Standalone"
     Platform="Windows" Runtime="XHTML" InstallFileSize="1024">
   <Wise Launch="1" ExecuteFile="C:\Program Files\Wise
         InstallMaster Demo\wise32.exe"
         InstallationDirectory="C:\TEST2" ImageFile="">
   </Wise>
  </CreateRuntimeInstallation>

  <CreateRuntimeInstallation DeploymentType="J2EE"
                           Server="weblogic;tomcat;websphere">
   <Libraries List="MODELS"/>
   <AdditionalFiles>
    <File Name="c:\temp\debug_1.log" Target="\WEB-INF\Lib" />
    <File Name="text2" Target="\WEB-INF\classes" />
   </AdditionalFiles>
  </CreateRuntimeInstallation>
 </OpenApplication>

 <Pack File="c:\temp\packtest.jpk" Name="TEST2" Libraries= ";"
                                        AdditionalFiles=";" />
 <Unpack File="c:\temp\myapp.jpk" Type="Java"          Target="MYAPP"
         InputDirectories="DDS;SDF;Screens"
         OutputDirectories="MakeExe;Runtime;Install"
         IncludeExtraFiles="True"  Existing="Replace"
         ConfigurationFiles="All">
```

```
 </Unpack>
</Ace>
```

For the complete DTD, which describes all possible attributes, refer to the `Buildapp.DTD`.

Logging

When running ACE in automatic mode, the following log is created: `remote.log` in the ACE root folder. The log begins with the contents of the `buildapp.xml` file. The `remote.log` reports the progress of the operations listed in the `buildapp.xml` file. In addition, information is also logged to the standard logs created by the Generate Runtime and Pack/Unpack Wizards.

Error Handling

When there is an an error in any of the operations in the file, processing will terminate immediately. This prevents the accumulation of several problems, one on top of the other. For instance, if generating the runtime fails, then creating a runtime installation might create an installation of the previous version.

The following table describes different types of errors that may occur, and how to handle them.

Table 14 - 60 : JIS 9.0B Command-Line ACE Error Resolution

Error Types	Reasons for Errors	What to Check
Syntax Errors	• Malformed XML (i.e. XML tags not closed) • Children tags appear outside of parent tags • Missing values in XML, missing attributes	• Verify the syntax of the XML • Verify that the XML conforms to the DTD.

Table 14 - 60 : JIS 9.0B Command-Line ACE Error Resolution

Error Types	Reasons for Errors	What to Check
Logical Errors	• Trying to generate a runtime that is not allowed for the application by the CDKey • Creating a runtime installation before the application was ever compiled	• Manually execute the same operations from the ACE UI.

We recommend performing the operations the first time manually, using ACE, in the same order as in the buildapp.xml, and verifying that they work properly, before using the automatic mode.

Limitations

The following known limitations exist:

- The feature is certified only for the following product flavors:
  - JIS XHTML for 5250
  - JIS Java for 5250
  - JIS XHTML for 3270
  - JIS Java for 3270
- The main window of ACE is still visible in automatic mode
- The operation *Create J2EE Runtime Installation* requires that you Generate Runtime for the application and all of its libraries at least once from within ACE. Otherwise, you get the following Perl error:

  ```
  "Error: Key 'libraries' not found in section 'program' at
  …\\perl\gen/HierarchyFile.pl line 69".
  ```

  Workaround:

  Generate the runtime in automatic mode, then edit the `<AppName>.ini` and add the following setting:

  ```
  [Program]
  Libraries=<semicolon separated list of libraries>;
  ```

  If you do not have any libraries, then the list should just contain the name of the application. For example, Libraries=TEST; for an application named TEST.
- Do not use comments (<!-This is a comment->) in the buildapp.xml file.

- When the packaging of the J2EE runtime installation fails, remote.log still reports successful completion.
- When using the command line interface, if you set the NewAndModified attribute to 1 the first time you generate runtime, the compilation fails. The first time you compile you must not use NewAndModified.

Improved User Interface

Introduction

The JIS Java Client user interface has been improved to create a more modern look and feel for the JIS Java components and to improve the table components functionality.

> **Note:** Excluding the mouse wheel support explained below, all of the other features are disabled by default in order to maintain backward compatibility.

To enable the new JIS look & feel add the following Applet parameter:

```
<PARAM name = "ThemeName" value = "default">
```

In addition JIS now offers the ability to customize the look & feel per project specific requirements. To achieve this, users will need to implement the UIManager interface or extend the DefaultTheme class.

Setting the following Applet parameters activates a user defined Theme class:

```
<PARAM name = "ThemeName" value = "UserDefined">
<PARAM name = "UserDefinedThemeClassName" value = "Fully qualified name of the Theme class">
```

To learn more about implementing project specific look & feel we recommend:

1. Reading the JavaDoc for interface cst.gwt.general.UIManager and for class cst.gwt.general.DefaultTheme in the JIS client Java Doc (located in ..\JacadaFiles\Docs\Client).
2. Reviewing the sample theme classes provided as part of the JIS samples (located in ..\JacadaFiles\samples\features\themes).

These user interface improvements include:

Mouse rollover mode

Controls such as arrow buttons, scrollbars, checkboxes and radio buttons will now change their look when the mouse rolls over them. This feature is only enabled when a Theme name is defined.

Mouse wheel support

Mouse wheel support is incorporated into window scrollbars and table scrollbars when using cached tables. This feature is only enabled when using JDK 1.4 and above.

Arrow buttons and scroll bars

The arrow buttons used in prompts, combo boxes, date fields and scroll bars are now displayed using a more modern look & feel.

Rounded rectangles for group boxes and frames

Group box, frames and tab headers now support “rounded corners”. This feature is only enabled when a Theme name is defined.

Calendar control

The calendar control’s look has been improved.

Client persistent storage

Changes to the table component including column resizing and reordering can now be saved locally on the user’s machine. Changes made by the user during the session lifetime will be saved to the disk when a session ends, and reloaded when a session starts. In addition, the user may save the current state using the **File->Save Storage** menu item and restore the default state using the **File->Clear Storage** menu items.

To enable the persistent storage feature add the following Applet parameter:

```
<PARAM name = "PersistentClientStorage" value = "true">
```

The persistent storage feature also requires working with the signed Applet and JDK 1.4 and above.

The persistent storage information is saved in a file named `JacadaClientProperties.xml` created in the user’s temporary files directory.

Table column reordering

This feature allows interactive table column reordering during runtime by using the mouse to drag and drop columns. In order to enable table columns reorder during runtime:

1. Make sure that the Drag columns checkbox is checked in the table control style tab in ACE
2. Add the following Applet parameter:

```
<PARAM name = "AllowTableColumnDragging" value = "true">
```

The column order initially defaults to the column order specified in ACE.

When the client persistent storage feature is enabled, the new column order is saved for the specific client machine.

Note the following limitations:

- Folded table columns cannot be reordered.
- When using fixed columns, the fixed columns cannot be reordered.

Table row sorting

When using fully cached tables, it's useful to sort the table data by column. This is now possible by clicking the column header of the column to be sorted. Clicking the column header once, sorts the table by ascending order, clicking again sorts the table by descending order. In order to return to the original host row order you must reload the table.

This feature is of limited use for most host tables since on normal tables it only sorts the current table page.

To activate this feature add the following Applet parameter:

```
<PARAM name = "AllowTableRowSorting" value = "true">
```

Sorting order is not maintained after exiting the current screen.

Known Limitations

- Heavy weight controls such as non transparent frames do not comply with the new look.
- Multiline labels are based on the java.awt TextArea component which draws its own scroll bars and therefore cannot be adapted to the JIS look and feel.

Additional Enhancements

Accessing the application server's HTTP session from an XHTML extension (ATL-27368)

It is now possible to access the application server's HTTP session from XHTML extensions:

```
/**
* Get J2EE sessionId
* @return sessionid
*/
 public String getSessionId ();
/**
* Binds an object to this session, using the name specified.
*/
public void setSessionAttribute(String name, Object value);
/**
* Returns the object bound with the specified name in this session,
* or null if no object is bound under the name.
*/
public Object getSessionAttribute(String name);
```

Loading resources from an XHTML extension (ATL-27368)

It is now possible to load resources from an XHTML extension:

Example 37. JIS 9.0B Loading Resources From an XHTML Extension

```
/**
* This method allows to read resource using ApplicationResourceLoader
* could be used in prop and J2EE server.
* @return InputStream
*/
public InputStream getResourceAsStream(String resource);
```

For example:

```
package appls.DEMO1.xhtml.user;

import com.jacada.jis.runtime.server.frontend.xhtml.context.*;
import
com.jacada.jis.runtime.server.frontend.xhtml.general.DocumentBuilderProvide
r;
import java.io.InputStream;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
```

```
import cst.debug.Debug;

public class Appl implements
com.jacada.jis.runtime.server.frontend.xhtml.extension.IUserPageExtension {

   public Appl () {}

   public void onPageLoad(OnPageLoadContext context) {
      Document staticXml = null;
      String xmlFilePath = "appls/"+ context.getLibraryName() +"/xml/"
+context.getSubApplName()+ ".xml";
      InputStream is = context.getResourceAsStream(xmlFilePath);
      try {
        staticXml = DocumentBuilderProvider.getDocumentBuilder().parse(is);
      } catch(Exception e) {}
      context.setSessionAttribute("staticXml", staticXml);
      // do some processing ...
      Document staticXmlFromSession =
(Document)context.getSessionAttribute("staticXml");
      NodeList nl = staticXmlFromSession.getChildNodes();
      Debug.print(1, "staticXml from HttpSession " + nl.item(0));
   }

   public void onPageSubmit(OnPageSubmitContext onSubmitContext) {}

}
```

Known Limitations

The new APIs cannot be used from `onPageSubmit()`.

XHTML Date control enhancements (EU-05423)

1 The calendar window default style:

   The default style for the calendar window can be overridden from an external CSS file (calendar.css). The calendar.css is created during runtime generation process, in the `appls/<APPLNAME>/xhtml/CSS`. By default, the file consists of only one line comment, to allow browsers to cache it.

   For example, if you want to increase the font size of the day numbers in the calendar window you can do it by defining a CSS definition:

```
.day font{
```

```
    font-size : 150%;
}
```

Following is a list of CSS class names in the calendar window:

- `currentDay` – class name for current day
- `weekendDay` – class name for weekend days
- `day` – class name for all other days
- `weekday` – class name for the titles of the week days(Sun, Mon)
- `currentDate` – class name for the current date title (August 2007)

2 It is now possible to use the "." character as a date separator

3 The default date format is selected according to the browser locale

Displaying digits in Java Client spin box controls (ATL-28026)

When you require adding a leading '0' in front of a single digit number in a spin box control, perform the following:

```
appls.<APPLNAME>.user.ApplSubApplwindow.java
public void setControl(Component comp, int tabIndex) {
         super.setControl(comp, tabIndex);
         if (comp instanceof GUISpin) {
             GUISpin spin = (GUISpin)comp;
             spin.setInputRestrictor(new NumberInputRestrictor("0#;"));
         }
}
```

Support for monochrome terminals (ATL-28319)

Added support for monochrome terminals. Set the following in the runtime-ini file to configure a model 5 monochrome terminal:

```
[GUISys TN5250]
TerminalType=IBM-3477-FG
```

SSL Connection (ATL-29082)

It is now possible to enable the SSL connection between the server and the host by simply configuring the cst.server.com.CSTSSLSocketFactory class to be the socket factory implementation. For example:

```
[GUISys TN5250]
SocketImplFactory= cst.server.com.CSTSSLSocketFactory.
```

Refer to "JIS 9.0B Appendix: Using Secured Telnet in JIS V9.0B" on page 310 for further details.

SSL connection limitations

The built-in CSTSSLSocketFactory solution does not support:

1. Client side of SSL encryption
2. ProxyServlet to server encryption
3. Server side of client to server SSL

It's only designed to solve the secured Telnet use case. All other combinations will still need to use extensions of the socket factory.

Changing XHTML Message Boxes (ATL-27839)

A new feature has been added which enables manipulating the text of a message box or completely eliminating the message box using an XHTML extension. For Example:

```
file src\appls\<app-name>\xhtml\user\Appl.java:
public void onPageLoad(OnPageLoadContext context) {
   Window window = context.getWindow();
   XhtmlControl control = window.getControlByName("messageBoxText");
   if (control != null) {
      if (control.getText().equalsIgnoreCase("Test message box")) {
         control.setText("");
      }
   }
}
```

This extension will remove the message box if it contains the text "Test message box" by changing its text to "".

Menus in XHTML (ATL-28670)

Infrastructure has been added to create a menu bar using in an XHTML extension.

JIS 9.0B Appendix: Using Secured Telnet in JIS V9.0B

Settings and Short Names

JacadaRootDir

This is the Jacada Root Directory, for example it could be c:\ace\JacadaFiles on a development machine. Typically, this is the folder that contains the classes folder and HTMLs.

JRELibDir

This folder is the lib folder for the JVM in use. It is `JacadaRootDir\utils\jre14\lib\`.

Important Files

CACERTS

This file is located in the **<**`JRELibDir`**>**`security` folder and is the Certificates Repository for the JVM in use. Any new-signed certificate should be imported into this cacerts file.

The default password for this store is '`changeit`'.

Secure Telnet between JIS and the Mainframe

Methodology

To enable the JIS support for Secure Telnet:

1. Obtain the secure telnet port on the Mainframe (or AS400) (Usually 992.) Open the `<AppName>.ini` and modify the Host IP address and port number to reflect these settings.
2. In the runtime-ini, add a new setting in the same section as the host and port, `SocketImplFactory= cst.server.comm.CSTSSLSocketFactory`.
3. The `<AppName>.ini` should now look like this:

```
[GUISys TN3270]
.............................
```

```
Port=992
Host=172.19.150.140
SocketImplFactory=cst.server.comm.CSTSSLSocketFactory
................
```

4. A certificate, generated by the security group for each mainframe and shared with the Jacada JIS server, is used to authenticate the Jacada JIS server with the Mainframe. Each mainframe should have a different certificate and these certificates may be changed periodically. Whenever a new certificate is generated, register the certificate file by importing it into the cacerts file in the `<JRELibDir>\security` folder.

   For Example:

   ```
   keytool -keystore .....\jre14\lib\security\cacerts -storepass changeit
   -import -file c:\temp\MainframeCert.cer
   ```

5. 5. Restart the JIS server.

You should now be able to connect to the JIS server and get valid responses from the Mainframe using Secure Telnet.

In previous versions of JIS it was necessary to manually implement the code below to create the SSL sockets. This is no longer necessary in V9.0B and above since this code is now part of the JIS `jacadasv.jar`

We still provide this code as reference and for customers using older versions:

Example 38. JIS 9.0B CSTSSLSocketFactory.java

```
package cst.sslSocket;

import java.net.*;
import cst.debug.*;
import java.security.KeyStore;
import javax.net.*;
import javax.net.ssl.*;
import javax.security.cert.X509Certificate;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;

public class CSTSSLSocketFactory implements cst.util.SocketFactory {

    SSLSocketFactory factory;

    public CSTSSLSocketFactory() {
        Debug.print(50, "CSTSSLSocketFactory()");
    }
```

```
    public java.net.ServerSocket createServerSocket(int port) throws
IOException {
        Debug.print(50, "CSTSSLSocketFactory 1::createServerSocket()");
        return null;
    }

    public java.net.ServerSocket createServerSocket(int port, InetAddress
bindAddr)
    throws IOException {
        Debug.print(50, "CSTSSLSocketFactory 2::createServerSocket()");
        return null;
    }

    public java.net.Socket createSocket(String host, int port)throws
IOException {
        Debug.print(50, "CSTSSLSocketFactory 8::createSocket()");
        if (factory == null) {
            factory = (SSLSocketFactory)SSLSocketFactory.getDefault();
        }
        Debug.print(50, "CSTSSLSocketFactory::after factory creation");
        Debug.print(50, "CSTSSLSocketFactory::before creating a normal
socket");
        Socket s = new Socket(host, port);
        Debug.print(50, "CSTSSLSocketFactory::after creating a normal
socket,
before creating SSL socket");
        Socket socket = (SSLSocket)factory.createSocket(s, host, port,
true);
        Debug.print(50, "CSTSSLSocketFactory::after createSocket() " +
socket);
        return socket;
    }
}
```

JIS Interface Server 9.0A07 Release Notes

New Features Included in JIS 9.0A07

The following new features have been added for JIS 9.0A07

- Refreshing the XHTML Client When a Page on the Host is Updated

Refreshing the XHTML Client When a Page on the Host is Updated

The client-oriented architecture of HTML means that requests always originate from the client. A state change on the host such that the host updates the server with a new screen was not displayed on the client unless the client sent a request to the host.

Session changes on the host are now "pushed" to the XHTML client, without the client requesting the updated server page. A new connection is opened to the server and if there is a screen change on the host, the new page is sent to the client immediately.

To enable pushing pages from the host to the client, in the `[XHTML]` section of the `<AppName>.ini` file set the `EnableAutomaticServerUpdates` parameter to 1.

> **Note:** Enabling this feature may impact significantly on scalability since it requires the client to maintain an open request (an open thread) to the server for the entire session.

JIS Interface Server 9.0A06 Release Notes

New Features Included in JIS 9.0A06

The following new features have been added for JIS 9.0A06:

- Platform Support
- Support for Keyboard Buffering

Platform Support

JIS proprietary server is now certified to run on AIX version 5.3 with IBM JRE 1.4.2 (32-bit).

> **Note:** To deploy to AIX version 5.3 requires the purchase of a new runtime license.Contact Jacada Services for the license.

Support for Keyboard Buffering

Many host emulators enable keyboard keystrokes to be buffered, so that users can continue typing without waiting for the host screen to refresh. After the host screen is refreshed, the content of the buffer is played back as if it was typed at that moment.

The Java client, keyboard buffering feature, requires an updated runtime license. Contact Software GmbH to obtain the runtime license file. Place the `license.dat` file in the `<JISRoot>\JacadaFiles\classes` folder.

Activating Keyboard Buffering

After installing the new license key that enables keyboard buffering, you must change parameters in the html file that launches the application, to activate the feature.

To activate keyboard buffering:

1. Open the html file that launches the application.
2. Change the `UseEventDispatchThread` parameter to a value "`true`" (if necessary, add the parameter):

```
<PARAM name="UseEventDispatchThread" value="true">
```

Setting this parameter true causes all server requests to be dispatched on the AWT event dispatch thread.

3 Make sure the `EnableKeyboardBuffering` parameter is set to true (the default value):

```
<PARAM name="EnableKeyboardBuffering" value="true">
```

4 Verify that keyboard buffering has been enabled by checking for the following debug print in the client log, set with debug level 50:

```
KeyboardBufferingManager is enabled
```

The following parameters influence how keyboard buffering functions.

Table 16 - 61 : JIS 9.0A06 Parameters for Keyboard Buffereing

Parameter	Default	Value Description
UseEventDispatchThread	true	You must set this parameter to true to use keyboard buffering. When set to true, all server requests are dispatched on the event AWT dispatch thread.
EnableKeyboardBuffering	true	This parameter must be set to true to use keyboard buffering. Set to false to turn off keyboard buffering.
KeyboardBufferingRestKey	Escape	The reset key code. Pressing the specified key resets the playback buffer. Valid codes can be obtained using: `KeyEvent.getKeyText(<Virtual key code>)`
KeyboardBufferingResetKeyModifier	Shift	Reset the key modifier. Valid codes can be obtained using: `KeyEvent.getKeyModifiersText(<Virtual modifier>)`

Table 16 - 61 : JIS 9.0A06 Parameters for Keyboard Buffereing

Parameter	Default	Value Description
HideKeyboardBufferingToolbar	true	Set this parameter to `false` to show the keyboard buffering toolbar when testing the application.

Known Limitations

The following limitations influence how keyboard buffering functions:

- The product's keyboard buffering feature must replace existing extensions that provide keyboard buffering functionality
- Replace every occurrence of JacadaStarter `activate(..., true)` API in the client extensions to JacadaStarter activate(..., false)
- Manipulating key events or focus events using Java extensions can adversely impact keyboard buffering
- Buffering key events starts after the display of the first window

JIS Interface Server 9.0A05 Release Notes

New Features Included in JIS 9.0A05

The following new features have been added for JIS 9.0A05

- API available to trigger server methods
- Printing

API available to trigger server methods

An API is now available to enable triggering a given method on the server directly from the Java client, without the need to trigger a menu item, button or accelerator linked to the method:

```
/**
* Activate method on the server by trigger id.
* The trigger id can be obtained as follows:
* For subapplication specific methods: from the
* <subapplication>.sa file
* For GUTMs: from applicat.ion
* @param id trigger id of the method as written in the
* applicat.ion or .sa file.
* @param wait true to lock the current thread's execution until
* the action is finished.
* @exception IllegalStateException if the method is invoked from
* the CommServer thread.
*/
public void activate(int id, boolean wait)
```

Printing

A number of changes were made to the way printing is handled.

Support for portrait and landscape printing

The Printer setup dialog for the Java and XHTML clients now supports printing both portrait and landscape printouts.

Enabling specifying margins

You can now specify the margins in the Printer setup dialog instead of using the default one inch margin.

Enabling specifying the paper size

You can specify the paper size in the Printer setup dialog or in the applet parameters, using the PrinterEmulationPaperType setting:

```
<PARAM name = "PrinterEmulationPaperType" value = "A4">
<PARAM name = "PrinterEmulationPaperType" value = "LETTER">
```

Improved handling of underlines in print output

Underlining in a print output now works. For example, characters are no longer misaligned.

Enabling printing to any printer

You can now print to any printer defined on the network and not just to the default printer.

New printer parameters

The following parameters have been added to the application INI file:

- `DynamicCalculateWidth`

  When set to 1, the server calculates the width of each page in the print job, resulting in a better layout of each page. Note that this may cause a non-homogeneous look to the whole job, since different pages may have different font sizes (due to a difference in the actual text width).
- `IgnoreEMAtStartOfLine` in the `[TN3270 section]`

  When set to 1, the printer emulation suppresses new line and carriage return characters, when they appear in the data stream, following the Mainframe “End of Medium” signal.

JIS Interface Server 9.0A02 Release Notes

New Features Included in JIS 9.0A02

The following new features have been added for JIS 9.0A02

- Enabling reconnecting to a database after the connection or session fails
- Limiting the size of the server log files

Enabling reconnecting to a database after the connection or session fails

When using external data methods to connect to a database, by default all sessions in a given server process, reuse the same JDBC connection. If this connection fails, every session using this connection may not be able to connect to the database.

To recover from a connection failure, the following new methods are available:

RepairDBConnection and RepairDBSession. The following example shows the usage of these methods:

```
Action: Query_with_close
Trigger: 17001 WaitIndicator: True
ScrambleName: False MoveMode: MoveNone
Description: ‘’
#0 = DoMethod: Receiver: ‘EXTERNALDATA’ Method: AllocDBSession Parms: (
‘"jdbc:as400://10.90.17.18;libraries=LY,*LIBL"’ , ‘"USER"’ , ‘"PSSWRD"’
)
If: Cond: ‘#0 == _FAIL ’
#0 = DoMethod: Receiver: ‘ExternalData’ Method: RepairDBConnection Parms:
( ‘"jdbc:as400://10.90.17.18;libraries=LY,*LIBL"’ , ‘"USER"’ ,
‘"PSSWRD"’ )
EndIf:
#3 = DoMethod: Receiver: ‘#0’ Method: ExecuteQuery Parms: ( ‘"select *
from sections"’ )
If: Cond: ‘#3 == -1’
#0 = DoMethod: Receiver: ‘ExternalData’ Method: RepairDBSession Parms: (
‘#0’ )
#3 = DoMethod: Receiver: ‘#0’ Method: ExecuteQuery Parms: ( ‘"select *
```

```
from sections"' )
EndIf:
Do: Times: '1000'
#4 = DoMethod: Receiver: '#0' Method: Next Parms: ( )
If: Cond: '#4 == _FALSE '
Break:
Else:
#5 = DoMethod: Receiver: '#0' Method: GetStringByColumnIndex Parms: ( '1'
)
DoMethod: Receiver: 'System' Method: DebugPrint Parms: ( '1' , '"row:" +
#5' )
EndIf:
EndDo:
DoMethod: Receiver: '#0' Method: Close Parms: ( )
```

Limiting the size of the server log files

The server can now create a new log file each time the size of the current log file exceeds a predetermined limit. The current log is renamed using a revision number.

The maximum size for the current log before a new log is created is set via the setting, RtDebugFileMaxSize. When `RtDebugFileMaxSize=0`, logging is always performed to a single log file, unlimited in size.

There is also a setting, for the maximum number of files that a server process is allowed to create. The `RtDebugMaxFiles` setting specifies the number of log files that the server can create.

These parameters can be set in the `jacadasv.ini` file or from the command line, as follows:

Via `jacadasv.ini` – Specify the parameters in the `[GeneralParameters]` section:

```
[GeneralParameters]
RtDebugFileMaxSize=<max_size_of_log_file_in_bytes>
RtDebugMaxFiles=<number of log files>
```

Via the standalone server command line – Specify the following switches for the parameters:

```
RtDebugFileMaxSize -m<max_size_of_log_file_in_bytes>
RtDebugMaxFiles -b<number of log files>
```

JIS Interface Server 9.0A01 Release Notes

New Features Included in JIS 9.0A01

The following new features have been added for JIS 9.0A01

- Enable opening a window in a maximized state
- Deploying a service to a J2EE Server

Enable opening a window in a maximized state

The Java Client now enables writing a Java extension for opening the Applet window in maximized state.

To maximize a window, use the following code:

```
package appls.TEST.user;

import java.awt.*;
import cst.gwt.*;

public class MainWindow extends appls.TEST.original.MainWindow {
   boolean firstShow = true;

   public void setVisible (boolean show) {
      if (show && firstShow) {
         firstShow = false;

((GUICSTFrame)getMyFrame()).setExtendedState(Frame.MAXIMIZED_BOTH);
      }
      super.setVisible(show);
   }
}
```

Register the `MainWindow.java` extension in the `JacadaStarter` using the `addWindow()` API

Deploying a service to a J2EE Server

The HTML page generated by JIS contains links to resources such as images, script files, css files, HTML files, etc. The URLs specified by these links differ between the standalone server and J2EE deployment. In the standalone server the URLs start with 'classes' (for example, src="/classes/js/jacada.js", href="/classes/appls/NRT/xhtml/CSS/kb_IE.css") while in J2EE deployment the URLs start with the name of the application (for example, src="/NRT/js/jacada.js", href="/NRT/appls/NRT/xhtml/CSS/kb_IE.css").

All URLs created by the server code are automatically adjusted, based on the server type (standalone or J2EE). However, URLs added through user extensions (java extensions, javascripts, htmls, etc.) should be changed as follows to ensure that the same code base will function properly for both the standalone server and a J2EE server.

In XHTML java extensions

Use an API method that returns the correct URL prefix: for the standalone server it returns /classes and for the J2EE server it returns /<APPLNAME>.

```
/**
* Get the application root dir
* @return String /classes - for standalone server, /<APPLNAME>
for J2EE Server
*/
public String getApplicationRootDir ();
```

In javascript extensions

Use javascript similar to the following:

```
window.addExternalJavaScriptFile(context.getApplicationRootDir() + "/appls/
TABLE/resources/test.js");
```

In the XHTML html extensions

Use the template word $ApplicationRootDir, which is changed to "/classes" or to "/<APPLNAME>", as follows:

```
<script language="JavaScript1.2" type="text/javascript"
src="$ApplicationRootDir/appls/TABLE/resources/test1.js"></script>
```

JIS Interface Server 9.0A00 Release Notes

New Features Included in JIS 9.0A00

The following new features have been added for JIS 9.0A00

- XINIT keyword in BMS maps now supported
- Maximum permitted size of ACE method increased
- Print setup dialog can be skipped
- New methods for setting colors of selected cells

XINIT keyword in BMS maps now supported

The screen image creation process now supports the presence of the `XINIT` keyword in a BMS file, without codepage considerations. The `XINIT` keyword in BMS maps allows the mapfield to be initialized to a hexadecimal value. A character with a value less than 0x40 is considered to be a terminal control character and is replaced with a blank.

Maximum permitted size of ACE method increased

The maximum number of lines in a single Ace method has been increased to 2048. Until now, the limit was 512 lines.

Print setup dialog can be skipped

When printing the current window, the Java client always shows the Printer Setup dialog, allowing the user to choose various printing options, including page type, page orientation, print destination, and so on.

You can now eliminate the display of the printer setup dialog if you so choose. To eliminate the display of the printer setup dialog, add the following parameter to the html page:

```
<PARAM name = "ShowPrintDialog" value = "false">
```

The default value is `"true"`.

New methods for setting colors of selected cells

New APIs have been added to the Java client `GUITable` component for setting the background and foreground colors of selected cells.

To set the background color in which selected cells are to be painted:

```
public void setSelectionBackground(Color color)
```

To set the foreground color in which selected cells are to be painted:

```
public void setSelectionForeground(Color color)
```

To discover if the default background color for selected cells has been changed (by `setSelectionBackground` or any other method):

```
public boolean isDefaultSelectionBackground()
```

To discover if the default foreground color for selected cells has been changed (by `setSelectionForeground` or any other method):

```
public boolean isDefaultSelectionForeground()
```

To discover the current background color of selected cells:

```
public Color getSelectionBackground()
```

To discover the current foreground color of selected cells:

```
public Color getSelectionForeground()
```

The following APIs are now marked deprecated since they only control the selection background color but not also the foreground color:

`public void setSelectionColor(Color color)` has been replaced by `setSelectionBackground (color color)`

`public boolean isDefaultSelectionColor()` has been replaced by `public boolean isDefaultSelectionBackground()`

`public Color getUserSelectionColor()` has been replaced by `public Color getSelectionBackground()`

A good place to use these APIs is by overriding the `setControl()` or `createGUIControls()` methods.

Example 39. JIS 9.0A00 Setting Cell Background Color

```
public void setControl(Component comp, int tabIndex) {
    super.setControl(comp, tabIndex);
    if (comp instanceof GUITable) {
        GUITable table = (GUITable)comp;
        table.setSelectionBackground(Color.blue);
    }
```

```
}
```